Designing and Implementing Skill Tests at Scale: Frequent, Computer-Based, Proctored Assessments with Minimal Infrastructure Requirements

Anastasiya Markova
University of California San Diego
La Jolla, USA
anmarkova@ucsd.edu

Anish Kasam
University of California San Diego
La Jolla, USA
akasam@ucsd.edu

Bryce Hackel
University of California San Diego
La Jolla, USA
bhackel@ucsd.edu

Marina Langlois
University of California San Diego
La Jolla, USA
malanglois@ucsd.edu

Sam Lau
University of California San Diego
La Jolla, USA
lau@ucsd.edu

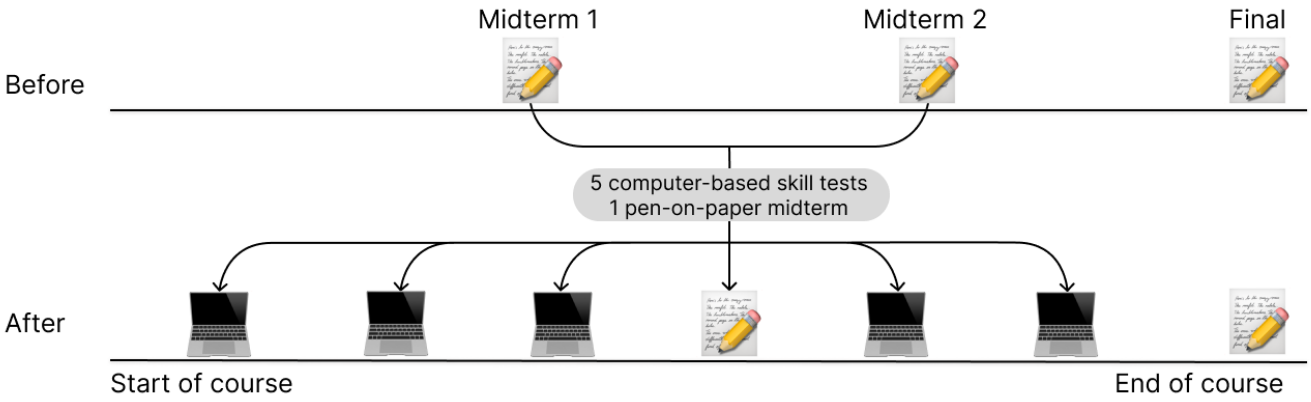


Figure 1: Our CS1 course historically had two pen-and-paper midterms and a final exam (Before). Now, one midterm has been replaced with 5 computer-based Skill Tests that take place throughout the course (After).

Abstract

The rise of Large Language Models has intensified the need for reliable assessments of foundational programming skills in introductory computer science courses. While frequent, low-stakes testing is an effective pedagogical strategy, its adoption is often slowed by the need for institutional infrastructure like a Computer-Based Testing Facility, which instructors may lack or find too inflexible. This experience report presents a practical, instructor-driven model for Skill Tests: weekly, 10-minute, proctored, computer-based assessments run by course staff in any campus room using student laptops. We provide a logistical blueprint refined over five offerings of a large CS1 course, detailing our strategies for question design, student scheduling, and staff management. Our findings show this model is effective: replacing a midterm with Skill Tests reduced student anxiety, with 91% of students preferring the new format. Student performance on the final exam remained consistent, indicating that knowledge synthesis was not compromised. This paper offers educators a framework for deploying frequent, low-stakes assessments without dedicated institutional resources.

CCS Concepts

• **Social and professional topics** → **Student assessment**.

Keywords

CS1, assessment, skill tests, frequent assessment, computer-based

ACM Reference Format:
Anastasiya Markova, Anish Kasam, Bryce Hackel, Marina Langlois, and Sam Lau. 2026. Designing and Implementing Skill Tests at Scale: Frequent, Computer-Based, Proctored Assessments with Minimal Infrastructure Requirements. In *Proceedings of the 57th ACM Technical Symposium on Computer Science Education V.1 (SIGCSE TS 2026), February 18–21, 2026, St. Louis, MO, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3770762.3772518

SIGCSE TS 2026, St. Louis, MO, USA

ACM ISBN 979-8-4007-2256-1/26/02
https://doi.org/10.1145/3770762.3772518

1 Introduction

The rise of Large Language Models (LLMs) like ChatGPT and GitHub Copilot has fundamentally challenged how introductory computer science (CS1) is taught and assessed, since these models

can easily solve typical introductory programming problems in both course assignments and summative assessments [2, 11]. To address this, computing instructors have adopted a variety of strategies to encourage students to use LLMs responsibly instead of adopting unhealthy habits of over-reliance, ranging from increasing the weight of exams on their course grade to developing "LLM-proof" assignments to even creating custom LLM tools for their courses with guardrails [7–9].

The authors of this experience report are also CS1 instructors who have needed to make substantial adjustments because of students using LLMs. After ChatGPT was released in 2022, we noticed a concerning trend. An increasing number of students earned perfect scores on take-home assignments but struggled to demonstrate the same knowledge in proctored settings. We speculated that this gap was due to an over-reliance on AI tools, which allowed students to bypass the deliberate practice that builds foundational programming knowledge. Similar to other instructors, we considered giving a higher weight to the course's exams and decreasing the weight on the take-home assignments [7]. This brought a long-standing issue to the forefront: high-stakes exams, already known to cause significant student anxiety and encourage cramming [4, 5], now needed to carry even more weight as one of the few reliable ways to gauge student ability.

One well-established approach to this issue is to increase the frequency of assessment. Shorter, more frequent, and lower-stakes assessments encourage continuous practice and can provide a more accurate picture of student learning [1, 3]. One practical way to implement this model is a Computer-Based Testing Facility (CBTF), which provides a proctored environment where students can take assessments with randomized questions on a flexible schedule [13, 16]. However, establishing a CBTF requires significant institutional support, funding, and physical space. Furthermore, even institutions with a CBTF may face logistical constraints. Our own institution, for example, has a CBTF run by a central campus unit that only offers 50-minute blocks, making it unsuitable for the 10-minute weekly quizzes we envisioned. This gap between the ideal of a CBTF and the reality on the ground motivates our work. What can instructors do if they want the pedagogical benefits of frequent assessment but lack the institutional infrastructure to support it?

In this experience report, we present a model for "Skill Tests": weekly, 10-minute, proctored quizzes that are run entirely by the course's own instructional staff using any available space on campus (we use small meeting rooms rather than classrooms). While this approach offers a path forward, it presents its own challenge: taking on the logistics of weekly testing for hundreds of students can be difficult, involving everything from scheduling spaces and staff to designing fair and secure questions.

The primary contribution of this experience report is not the idea of frequent testing itself, but rather a practical and tested blueprint for its implementation. We share the design of our Skill Test system, which we have deployed and refined over five offerings of a CS1 course of 150–300 students. We share the key logistical and pedagogical decisions that have made this model successful and sustainable, including our strategies for question design, student sign-ups, and efficient proctoring by course staff. Our goal is to provide a guide for other instructors, sharing lessons learned so they can adopt a similar model without having to reinvent the process from scratch.

We provide evidence that this instructor-driven model is both effective and beneficial. In response to concerns about logistics and assessment quality, we show that: 1) Rather than raising anxiety, the Skill Test format significantly reduced self-reported student anxiety, with over 85% of students preferring it to a traditional two-midterm structure. 2) Replacing a midterm with granular skill assessments did not harm students' ability to synthesize concepts, as final exam scores remained consistent after the introduction of Skill Tests. Our findings present Skill Tests as a viable and beneficial alternative to traditional midterm exams in CS1, particularly for instructors seeking to implement frequent, proctored assessments without reliance on a dedicated campus facility.

2 Related Work

In this section, we describe two bodies of research that motivated us to design Skill Tests: the pedagogical benefits of frequent assessment, and computer-based testing as a technological solution to logistical challenges of frequent assessment.

2.1 Benefits of Frequent Assessment

The core pedagogical principle supporting frequent assessment is the "testing effect", a well-documented phenomenon that refers to the finding that taking a test on studied material improves subsequent learning and retention [3], even more so than extra studying [10]. The testing effect has proven to be a robust phenomenon, replicated across numerous domains, educational contexts, and types of learning materials [6, 14]. Recent work in CS education has also found frequent assessment effective for teaching programming. For example, Smith et al. conducted a study in a CS1 course comparing a frequent testing regimen (weekly quizzes and smaller exams) with an infrequent one (a single midterm and final). They found that the frequently tested cohort significantly outperformed their peers on identical code-writing questions [12]. Moreover, students in the frequent-testing condition reported that the regimen encouraged better study habits, such as less cramming, and reduced their test anxiety.

2.2 Computer-Based Testing Facilities

Despite the overwhelming evidence of its pedagogical effectiveness, frequent assessment also imposes logistical burdens on instructors and teaching staff. Administering weekly assessments involves a significant recurring investment of time and resources, especially in the large-enrollment courses common for CS1, since course staff are responsible for preparing, proctoring, and grading the assessments. Computer-based testing facilities (CBTFs) have emerged as a solution to this challenge by providing proctored computer labs designed exclusively for administering assessments [15]. One institution's CBTF administers 50,000 exams per semester, demonstrating that this approach is scalable [16]. The primary disadvantage of the CBTF model is the significant upfront and ongoing investment required. This includes the physical space, dedicated computer hardware, servers, and, most significantly, a centrally-managed staff of trained proctors [13]. While this model can be cost-effective at scale, the high initial investment makes it inaccessible for many

institutions without substantial, sustained institutional support. We see Skill Tests as a lighter-weight way of capturing the benefits of frequent assessment for CS courses that do not yet have a CBTF that meets their assessment needs. Since Skill Tests only require coordination between course staff rather than institutional support, they provide a more accessible alternative.

3 Course Setting

The authors of this paper are instructors and course staff at a large, public, PhD-granting university in North America that operates on a quarter system. We teach an introductory programming course in Python required for data science majors. The curriculum includes programming fundamentals: variables, conditionals, loops, functions, recursion, time complexity, object-oriented programming, and Python's built-in data structures (lists, dictionaries, sets). Additionally, the course covers Python-specific language features such as lambda functions and list comprehensions. Enrollment varies by term, with approximately 150 students during the fall and spring quarters, and around 300 students during the winter quarter. Most students enter the course with minimal prior programming experience.

Our course runs for 10 weeks, with three 50-minute lectures per week taught by faculty and one 50-minute discussion section taught by course staff. Lectures and discussions use a combination of slides and live coding demonstrations. Additionally, students complete two programming assignments each week: a lab with simpler exercises and a homework with more challenging problems. Both assignments include hidden test cases, and homeworks are additionally graded for code style. On average, our department allocates 20 hours per week of graduate teaching assistant time for every 60 students, or alternatively 10 hours per week of undergraduate tutor time for every 15 students.

Historically, our course's assessment structure consisted of two midterms (in weeks 4 and 7) and a final exam during week 11. All exams were pen-and-paper, requiring students to write code by hand. A redemption policy allowed students to replace their midterm scores with their final exam score if it resulted in a higher grade. To implement Skill Tests, we replaced one midterm, originally worth 15% of a student's overall grade, with five Skill Tests, each worth 3%. The total weight of summative assessments (midterms, Skill Tests, and the final exam) remained at 50% of the final grade, consistent with the previous course structure.

4 The Design and Implementation of Skill Tests

This section details our approach to implementing Skill Tests. We structure our discussion into two parts. First, we describe the pedagogical principles that guide the design of the Skill Test questions themselves. Second, we outline the logistics of how we conduct the tests, from the student's perspective to the proctoring and support systems we have in place.

4.1 Designing Skill Test Questions

The core philosophy behind our Skill Tests is that they should assess students' grasp of concepts recently covered in lectures and labs. Each Skill Test consists of four questions that are designed to be close variations of problems students have already seen and practiced. This approach is intended to reward students for keeping up with the course material and to reduce anxiety associated with "trick questions." The focus is on verifying that students have achieved a baseline competency with a specific skill, not on testing their ingenuity under time constraints.

Table 1: Skill Test Topics

Skill Test	Topic
1	For Loops, While Loops, Conditionals
2	Dictionaries
3	List Comprehensions, Asserts
4	Map, Filter, Lambda
5	*args, **kwargs, Recursion

4.1.1 Question Topics and Format. To ensure that Skill Tests are focused, each test covers a small number of topics, and each Skill Test question assesses knowledge of a single topic rather than combining concepts together. At the start of the quarter, the instructor selects key concepts from the course to assess. Table 1 provides an overview of the topics covered by Skill Tests in our CS1 course.

Currently, all questions use a "write a function" format, where students must implement a function to match a given specification, though we are exploring other auto-gradable formats like Parsons Problems. An example question is shown in Figure 3.

4.1.2 Grading and Feedback. To provide immediate feedback and enable auto-grading, each question is accompanied by a set of unit tests. Typically, we provide two public tests (visible to students during the test) and two hidden tests (used for final grading). Students receive credit for each hidden test passed. The tests are designed to check for correct implementation of the core concept and deliberately avoid edge cases (e.g., empty inputs or inputs of the wrong type), reinforcing the focus on foundational understanding. The public tests allow students to debug their code during the assessment, mirroring a more authentic programming process.

4.1.3 Development and Versioning. Four new questions are collaboratively developed by the course staff for each Skill Test. Once the topics are decided, staff members draft questions, which then undergo review. Senior staff members check for clarity and correctness, while junior staff members beta-test the questions to ensure the difficulty is reasonable for the 10-minute time frame. To maintain academic integrity, we create two versions of each Skill Test. Version 1 is administered on the first testing day, and Version 2 is used on the second. We have found that graduate and undergraduate course staff members can draft high-quality questions using this workflow with minimal oversight from the faculty instructor. However, we would recommend that other instructors trying Skill Tests for the first time play a more hands-on role in reviewing questions to calibrate difficulty.

4.2 The Logistics of Conducting Skill Tests

Running weekly assessments for hundreds of students requires a robust logistical framework. This section outlines our process, from the student's initial sign-up to our proctoring strategy and support systems.

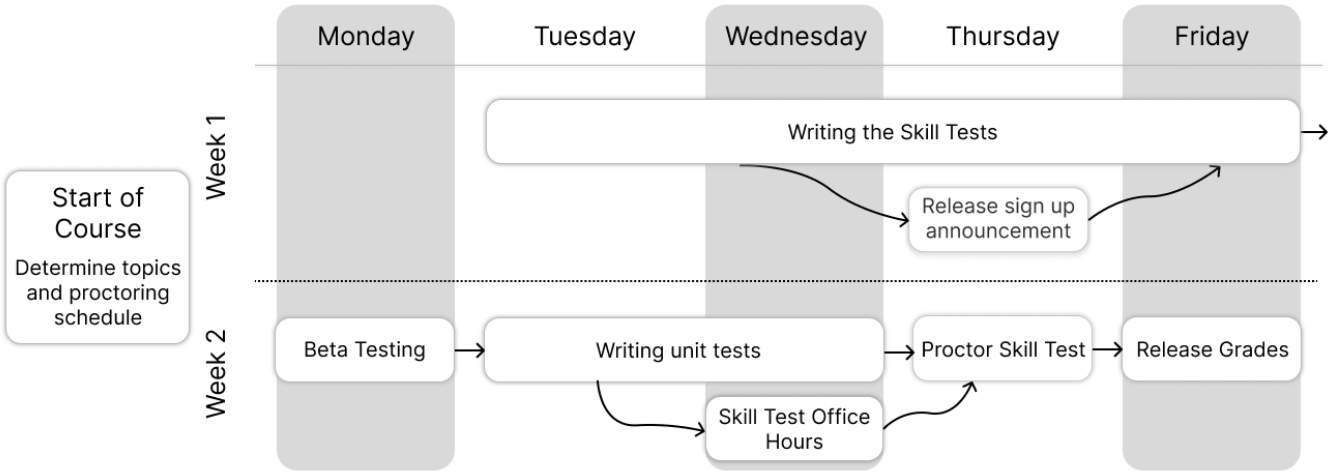


Figure 2: A timeline of the development process for a Skill Test in our CS1 course, which spans two weeks. Course staff draft Skill Test questions, beta test them, and write unit tests before offering the Skill Test to students.

```
def mean_price(cart):
    '''
    Function that calculates mean price of items in cart

    args:
        cart (dict): dictionary containing foods and prices
    returns:
        average cost of items in the cart

    >>> cart = {"steak": 14, "milk": 5}
    >>> mean_price(cart)
    9.5
    >>> cart = {"chicken": 8, "juice": 7, "eggs": 6}
    >>> mean_price(cart)
    7.0
    '''
    # YOUR CODE HERE
```

Figure 3: Example skill test question on accessing dictionary keys and values. Students solve 4 such questions during the 10-minute period. Hidden tests closely mirror doctests and do not test edge cases.

4.2.1 The Student's Perspective. Students first learn about Skill Tests from the course syllabus and an announcement during lecture. One week before each test, an announcement on the course discussion board provides key details: a list of the topics covered, question format, sign-up instructions, and links to practice problems. Students sign up for a 15-minute time slot (10 minutes for the test, plus 5 for setup) to minimize wait times. On testing day, students arrive at their scheduled time, present their student ID to a proctor, and receive a link to the test. During the test, students can write, run, and submit code, but are not allowed to use other web pages or applications. Once all testing for the week is concluded, students can view their grades, submitted code, and test case results on the course's grading platform.

4.2.2 Scheduling, Sign-ups, and Retakes. To ensure academic integrity, each proctor oversees a small group of 3–4 students. We calculate the total number of proctoring hours needed per week with the following formula:

$$\text{Proctor Hours} = \frac{(\text{Class size}) \times (\text{Scheduling overhead})}{(\text{Students per group}) \times (\text{Groups per hour})}$$

The scheduling overhead term refers to creating more time slots than are strictly necessary to ensure students have flexibility when scheduling. In our course, we use an overhead of 1.2 (i.e., we offer 20% more slots than are strictly required), which prevents students who sign up later from having very few available options. With 15-minute time slots (10 for the test, 5 for transition), a single proctor can handle 4 groups per hour. The total hours are then distributed across several testing windows on different days to accommodate student schedules. To give a concrete example, for the Spring 2025 offering of our CS1 course with 107 students where each proctor managed 3 students at once, we allocated 11 proctor hours per week, which we distributed evenly among our course staff.

Our sign-up policy is flexible. If students need to reschedule, they are instructed to drop into any available testing session on the day of the Skill Test without needing to sign up again or contact course staff.

We also offer a single retake at the end of the quarter. For the retake, we select four questions from the previous five Skill Tests, typically choosing those with the lowest average scores. All students receive the same set of retake questions, and the score can replace their lowest Skill Test grade.

4.2.3 Software and Administration. We use web-based platforms that allow students to write and execute code in the browser. The essential software requirement is the ability for students to run their code against public test cases during the assessment, which helps them debug their work. The platform also manages the 10-minute time limit and auto-submits the student's work when time expires.

4.2.4 Proctoring and Student Support. Tests are conducted in small meeting rooms on campus. To manage the 3–4 student to 1 proctor ratio, proctors are seated behind the students they are responsible for, allowing them to monitor screens while maintaining a quiet environment.

To help students prepare, we offer dedicated Skill Test office hours. These sessions begin with a review of the week's topics, followed by a live, step-by-step walkthrough of practice problems created by the course staff.

4.2.5 Lessons Learned. We found that small group, in-person proctoring was essential for efficiency and integrity. In our experience, when proctors needed to oversee more than 4 students, our proctors found it difficult to maintain our academic integrity standards, which is why we converged on 3–4 students per proctor. Dedicated office hours for Skill Tests were also highly popular, as students were motivated to attend and learn the material they knew would be on the upcoming assessment. For instance, in Spring 2025, a typical office hour had between 1–2 students weekly, while a Skill Test office hour had between 10–15 students.

Initially, we called these assessments "Skill Demos", but this was problematic; it suggested a low-stakes exercise, creating a mismatch with student expectations. Renaming them to "Skill Tests" better conveyed their nature as assessments. Similarly, an early, rigid sign-up system created friction. A flexible scheduling policy that allowed students to drop into any available session worked much better. Finally, we learned to scope questions carefully. We found that topics requiring significant boilerplate, like file reading and writing, were ill-suited for a short, timed assessment as students struggled to recall syntax under pressure.

5 Student Outcomes

To understand the effectiveness and impact of our implementation of skill tests, we analyzed results from three perspectives. First, we directly analyze skill test scores, looking at factors like student completion rate, median, and standard deviation. We then analyzed final exam scores, trying to determine if the introduction of Skill Tests affected student performance in a typical exam setting. Finally, we analyzed responses from an end-of-quarter survey to understand student stress and confidence levels, along with overall experience feedback.

5.1 Skill Test Scores

Student scores on skill tests from Spring 2025 are displayed in Figure 4. Across Skill Tests 1–5, between 52% and 78% of students scored in the top band (1.5–2.0), and the medians were at least 1.5 (with Tests 3 and 4 at 2.0). Since the test cases used are not edge cases, a full score corresponds to finishing the test on time. In other words, most students score above 80% on skill tests, which suggests to us that the 10-minute time limit is generally adequate for most students.

5.2 Final Exam Scores

To evaluate whether replacing a midterm with Skill Tests affected students' ability to synthesize knowledge, we compared final exam scores across eight offerings of the course. Figure 5 displays the average final exam scores for three offerings before this change (Winter 2023–Fall 2023) and five offerings after (Winter 2024–Spring 2025). The data shows that final exam scores remained stable, with no significant change in performance after the introduction of Skill Tests. This suggests that the shift to more frequent, focused assessments did not compromise students' performance on a comprehensive final exam.

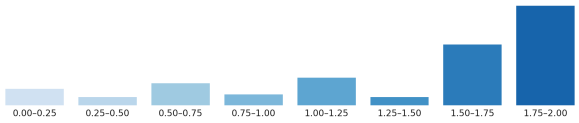


Figure 4: Skill test scores from Spring 2025. Most students complete the questions within the allotted time.

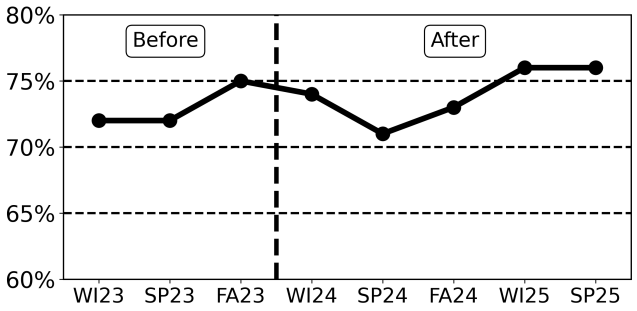


Figure 5: Final exam scores stayed consistent after the introduction of Skill Tests in Winter 2024.

To further investigate, we analyzed performance on final exam questions that covered topics directly assessed in Skill Tests, such as list comprehensions, dictionary manipulation, and recursion. Here too, we found no substantial difference in student scores between the offerings with and without Skill Tests. However, direct conclusions are difficult to draw because of factors like exam question changes and curving policies. More research is needed to fully understand the impact on final exam scores.

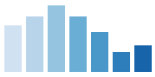
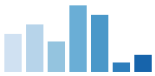
Note that this analysis has limitations which prevent us from conducting statistical tests. Namely, final exams differed between quarters, though they were written by the same instructor. In addition, student populations vary by quarter; spring offerings, for instance, tend to include more non-majors who score lower on average.

5.3 Student Survey

To assess student perceptions of Skill Tests, we administered an optional, end-of-quarter survey for extra credit in Spring 2025. We received 79 responses from a class of 107 students (74% response rate).

5.3.1 Student Anxiety. To gauge student anxiety, we asked students to rate their stress levels on a 7-point scale in response to different course-related scenarios. As Table 2 shows, the mean reported anxiety for a Skill Test (3.6/7) was substantially lower than for a midterm exam (5.4/7), and only slightly higher than the anxiety of a homework deadline (3.3/7). These results suggest that the frequent,

Table 2: Students generally reported less anxiety before Skill Tests compared to midterms, as reported in an end-of-quarter survey. Rating scale from left-to-right ranged from ▫ = Not Anxious at all (1) ; to ■ Very Anxious (7).

Scenario	Distribution	Mean
The night a homework is released		2.1 / 7
The night a homework is due		3.3 / 7
Right before a Skill Test		3.6 / 7
Right before a Midterm		5.4 / 7

low-stakes format of Skill Tests created a less stressful experience for students than traditional high-stakes exams.

5.3.2 Student Confidence. Skill Tests appeared to boost student confidence. When asked to respond on a 5-point Likert scale to "I feel more confident working on labs/homeworks after taking the skill tests", 61.7% of respondents agreed that the weekly assessments increased their confidence in their ability to complete homework assignments and perform well on exams.

5.3.3 Format Preferences. Students strongly preferred the Skill Test format over the traditional midterm structure. When asked whether they would prefer one midterm and weekly Skill Tests or two midterms, 91% of students chose the Skill Test option. However, it should be noted that the students surveyed weren't able to choose between formats in this course. More research is needed to explore the sentiment of students when given the choice between these two options.

5.3.4 Overall Sentiment. Qualitative feedback was largely positive. Students frequently commented that they appreciated the low-stakes, straightforward nature of the tests. Many also noted that the weekly cadence helped them keep pace with the course material, describing the tests as a useful "check for understanding." The dedicated Skill Test Office Hours were also highly valued—students who attended described them as a positive experience that helped "fill holes in their understanding." However, students also pointed out logistical drawbacks of Skill Tests, noting that the 10-minute time limit to complete the assessment felt too short at times, and the requirement to attend in-person sessions could sometimes be burdensome (e.g., for students who commuted to campus).

6 Discussion

In this section, we reflect on the strengths and limitations of our Skill Test approach, its role alongside our campus-provided CBTF, and implications for future assessments in light of student usage of GenAI tools.

6.1 Strengths and Limitations of Skill Tests

When we first launched Skill Tests, we were concerned that students would feel more anxious because of the more frequent pressure to perform well on an assessment. Instead, we found that students generally felt less anxiety about Skill Tests and strongly preferred Skill Tests over traditional midterms, as reported in Section 5.3. We were also concerned that replacing a midterm with granular, weekly quizzes would discourage students from developing the ability to synthesize multiple concepts. However, our data show that this was not the case; final exam scores, which assess complex problem-solving, remained consistent after the introduction of Skill Tests. This structure also appeared to encourage students to engage more deeply with the material themselves; when asked for advice to future students, many survey respondents recommended completing assignments without relying on tools like ChatGPT.

That said, running weekly assessments still requires a substantial investment of course staff time for creating questions, test cases, and proctoring. For our course, we felt that a student-to-staff ratio of 4:1 was necessary to maintain academic integrity since students were using their personal laptops. For other institutions, particularly those without a larger pool of undergraduate course staff, this resource cost could be a major barrier. This represents a clear opportunity cost: staff time spent on proctoring is time that cannot be spent on other valuable activities, such as holding more office hours or providing more detailed feedback on assignments. We recommend that other instructors carefully evaluate whether allocating staff time to run Skill Tests is worth it.

6.2 Skill Tests vs. CBTFs

We see Skill Tests not as a replacement for using a CBTF but rather as a pragmatic alternative for scenarios where using a CBTF is infeasible. For example, at our institution, it took years of effort from both faculty and staff to muster enough support to establish our CBTF. Although we would ideally administer Skill Tests through our CBTF, we are currently unable to do so because our CBTF is managed by a central campus unit and currently only offers inflexible 50-minute timeslots. We expect that other instructors may run into similar challenges, whether they have yet to launch a CBTF or whether their CBTF does not yet offer enough flexibility for their course needs.

In this way, Skill Tests can act as a "bridge" for instructors who want the benefits of frequent testing immediately even without access to a CBTF. Skill Tests can be implemented by a single course's staff using any available campus space. For example, to run Skill Tests, we book meeting rooms in our department's building that hold between 8–16 students at a time; these rooms are easier to reserve than general-purpose classrooms since they are managed by our department. While we still hope to migrate our Skill Tests to our campus CBTF once it offers the necessary logistical flexibility, we believe our model provides a viable and effective path forward for the many instructors who do not have that option.

References

[1] Robert L. Bangert-Drowns, James A. Kulik, and Chen-Lin C. Kulik. 1991. Effects of Frequent Classroom Testing. 85, 2 (1991), 89–99. doi:10.1080/00220671.1991.10702818

[2] Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. 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 V. 1* (2023). 500–506.

[3] Andrew C. Butler and Henry L. Roediger. 2007. Testing Improves Long-Term Retention in a Simulated Classroom Setting. 19, 4–5 (2007), 514–527. doi:10.1080/09541440701326097

[4] John J. Donovan and David J. Radosevich. 1999. A Meta-Analytic Review of the Distribution of Practice Effect: Now You See It, Now You Don't. 84, 5 (1999), 795. https://psycnet.apa.org/record/1999-01454-012

[5] Ray Hembree. 1988. Correlates, Causes, Effects, and Treatment of Test Anxiety. 58, 1 (1988), 47–77. doi:10.3102/00346543058001047

[6] Robert M. Hogan and Walter Kintsch. 1971. Differential Effects of Study and Test Trials on Long-Term Recognition and Recall. 10, 5 (1971), 562–567. https://www.sciencedirect.com/science/article/pii/S0022537171800294

[7] Sam Lau and Philip Guo. 2023. 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 V.1* (Chicago IL USA, 2023-08-07). ACM, 106–121. doi:10.1145/3568813.3600138

[8] Bradley McDanel and Ed Novak. 2025. Designing LLM-Resistant Programming Assignments: Insights and Strategies for CS Educators. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (Pittsburgh PA USA, 2025-02-12). ACM, 756–762. doi:10.1145/3641554.3701872

[9] James Prather, Juho Leinonen, Natalie Kiesler, Jamie Gorson Benario, Sam Lau, Stephen MacNeil, Narges Norouzi, Simone Opel, Vee Pettit, Leo Porter, Brent N. Reeves, Jaromir Savelka, David H. Smith, Sven Strickroth, and Daniel Zingaro. 2025. Beyond the Hype: A Comprehensive Review of Current Trends in Generative AI Research, Teaching Practices, and Tools. In *2024 Working Group Reports on Innovation and Technology in Computer Science Education* (Milan Italy, 2025-01-22). ACM, 300–338. doi:10.1145/3689187.3709614

[10] Henry L. Roediger and Jeffrey D. Karpicke. 2006. Test-Enhanced Learning: Taking Memory Tests Improves Long-Term Retention. 17, 3 (2006), 249–255. doi:10.1111/j.1467-9280.2006.01693.x

[11] Jaromir Savelka, Arav Agarwal, Marshall An, Chris Bogart, and Majd Sakr. 2023. Thrilled by Your Progress! Large Language Models (GPT-4) No Longer Struggle to Pass Assessments in Higher Education Programming Courses. In *Proceedings of the 2023 ACM Conference on International Computing Education Research V.1* (Chicago IL USA, 2023-08-07). ACM, 78–92. doi:10.1145/3568813.3600142

[12] David H. Smith, Chinedu Emeka, Max Fowler, Matthew West, and Craig Zilles. 2023. Investigating the Effects of Testing Frequency on Programming Performance and Students' Behavior. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON Canada, 2023-03-02). ACM, 757–763. doi:10.1145/3545945.3569821

[13] Jim Sosnowski, Julie M. Baker, Olivia Arnold, Mariana Silva, David Mussulman, Craig Zilles, and Matthew West. 2024. Reflections on 10 Years of Operating a Computer-based Testing Facility: Lessons Learned, Best Practices. In *2024 ASEE Annual Conference & Exposition* (2024). https://peer.asee.org/47930.pdf

[14] Mark A. Wheeler and Henry L. Roediger. 1992. Disparate Effects of Repeated Testing: Reconciling Ballard's (1913) and Bartlett's (1932) Results. 3, 4 (1992), 240–246. doi:10.1111/j.1467-9280.1992.tb00036.x

[15] Craig Zilles, Robert Timothy Deloatch, Jacob Bailey, Bhuwan B. Khattar, Wade Fagen-Ulmschneider, Cinda Heeren, David Mussulman, and Matthew West. 2015. Computerized Testing: A Vision and Initial Experiences. In *2015 ASEE Annual Conference & Exposition* (2015). 26–387. https://peer.asee.org/computerized-testing-a-vision-and-initial-experiences

[16] Craig B. Zilles, Matthew West, Geoffrey L. Herman, and Timothy Bretl. 2019. Every University Should Have a Computer-Based Testing Facility.. In *CSEDU (1)* (2019). 414–420. https://pdfs.semanticscholar.org/6eb4/544d8cb71e1bb1a0659b18d3c5f327e596c9.pdf