2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #52430

Empowering Student Agency and Engagement through Gamified Debugging and Flexible Assessment Design in Introductory Programming

Ekincan Ufuktepe, University of Missouri-Columbia

Ekincan (Ekin) Ufuktepe is an Associate Teaching Professor, and the Director of Undergraduate Studies at the Electrical Engineering and Computer Science department at the University of Missouri-Columbia. He has received many teaching and mentoring awards at the department, college, and campus levels. He has been teaching introductory and core computer science and computer engineering classes for more than a decade. Lately, he has been teaching ”Algorithm Design and Programming I”, which is an introductory class primarily offered to computer science, computer engineering, and electrical engineering students, where enrollment goes up to ˜350 students per semester. Due to the large enrollment and the class being an introductory class, it brings various challenges, which are mostly student engagement, student anxiety, attendance, and participation. Therefore, every semester, Ekin has developed and experimented with various techniques and strategies to engage students, reduce anxiety, and improve attendance and performance.

Deniz Kavzak Ufuktepe, University of Missouri - Columbia
James Ries, University of Missouri - Columbia

Empowering Student Agency and Engagement through Gamified Debugging and Flexible Assessment Design in Introductory Programming

Abstract

This paper presents two complementary pedagogical innovations implemented in an introductory programming course (Algorithm Design and Programming I) to enhance student engagement, metacognitive reflection, and ownership of learning. The first innovation, "Bounty Hunting/Pest Control," which is actually debugging sessions, consists of short, gamified exercises conducted weekly at the start of class, where students work collaboratively in pairs or small groups to identify and fix intentional bugs in concise code snippets aligned with recent lecture topics. Successful participants earn symbolic "bounties" recorded, with small, non-grade incentives such as class-branded items awarded at semester's end. This approach promotes authentic, low-stakes problem-solving, normalizes debugging as a core learning process, and fosters a sense of community and motivation without grade pressure. The second innovation, a Grade Weight Adjustment Policy, provides students with structured flexibility in determining the relative weights of course components (labs, exams, and quizzes) within bounded ranges at the end of the semester. This design empowers students to reflect on their strengths, mitigate anxiety associated with assessment, and perceive the grading process as more transparent and equitable. Early feedback suggests these interventions have led to higher participation, improved self-assessment, and greater confidence in programming skills. Together, the two strategies create a learner-centered environment that integrates autonomy, active engagement, and self-regulation, which are key components of effective engineering education. These practices demonstrate how gamification and flexible assessment can coexist within rigorous, large-enrollment STEM courses to strengthen both skill development and student agency, offering an adaptable model for other engineering and computing programs.

Introduction

In large introductory programming courses, keeping students engaged and helping them develop effective learning strategies is a persistent challenge. Novice programmers often struggle with debugging and can find it tedious or intimidating, leading to lower motivation and shallow engagement [1]. Traditional assessment schemes in such courses may also contribute to anxiety and a lack of student ownership over learning. To address these issues, we implemented two complementary pedagogical strategies in an introductory programming course (Algorithm Design and Programming I) aimed at increasing student engagement, metacognitive reflection, and sense of ownership: (1) "Bounty Hunting" gamified debugging sessions, and (2) a Flexible Grade Weight

Adjustment Policy. This paper describes the design of these interventions, situates them in relevant educational theory, and presents initial evidence of their effectiveness based on student feedback and performance data. We also discuss how these strategies align with established frameworks in gamification, metacognition, and assessment design, and how they can be adapted in similar educational contexts.

Related Work

Debugging is a fundamental but often frustrating skill for novice programmers. Prior work has explored gamification as a means to make debugging practice more engaging for beginners [2]. Educational games can increase student motivation and lead to deeper learning in computing courses. For example, Gidget [3] is a debugging-first puzzle game that requires novices to fix faulty code; studies showed that after only about five hours of gameplay, learners were able to grasp programming constructs like loops and conditionals. This suggests that a well-designed debugging game can effectively teach core concepts in a short time. Similarly, the RoboBug [4, 5] serious game was designed to drill debugging strategies in a fun, time-pressured environment, providing novices with guided practice using debugging tools. Studies of such games report improvements in students' debugging performance after play, indicating that gamified approaches can enhance both engagement and skill acquisition in early programming courses [2]. Beyond individual tools, broader evaluations of gamified learning in CS1 have generally found positive impacts on student engagement and skill development. By turning code troubleshooting into a low-stakes "game", these interventions help students view debugging as a fun challenge rather than a dreaded obstacle. Not only do competitive or narrative elements increase enthusiasm, but immediate feedback and rewards in these games build confidence and competence, which addresses factors known to influence intrinsic motivation. In summary, gamified debugging environments leverage the motivational power of games to improve novice programmers' willingness to practice debugging and, ultimately, their debugging proficiency.

Another strand of relevant work examines flexible grading policies that give students some choice in how their performance is evaluated. Traditional rigid weighting of exams, labs, and assignments can contribute to student anxiety and a lack of ownership over learning. In response, researchers have proposed flexible assessment weighting schemes where students can influence or select the weight of various course components. Evidence from higher education suggests that introducing such flexibility can empower students without compromising academic standards. In a recent study, Coyne and Woodruff [6] allowed undergraduates to choose their own assessment weightings and found no significant grade inflation or performance decline as a result. Students reported that this choice made the course more meaningful, satisfying their desire for greater control over their learning experience. These findings align with prior observations that giving students a voice in assessments can increase their sense of fairness and reduce stress. For instance, Wanner and Palmer [7] reported that students highly value having options in how they are assessed, although too many choices can be overwhelming. Similarly, Rideout [8] and others have documented that flexible assessment regimes can personalize the learning experience and help students play to their strengths. In engineering and computing education, structured flexibility, for example, allowing students to allocate different weights to exams vs. projects within set bounds, has yielded a "*happier, more engaged cohort*" without eroding rigor. Such schemes not only let students mitigate

weaknesses (for example, placing less weight on an exam if they excel in projects) but also encourage them to reflect on their own abilities when deciding weightings. This act of deliberating on one's strengths and weaknesses is itself a learning exercise. In summary, flexible grading approaches in introductory courses have been shown to maintain academic outcomes while enhancing student agency, thereby potentially improving motivation and reducing assessment-related anxiety.

Metacognition, or students' awareness and regulation of their own learning processes, is crucial in programming and engineering education. Novice programmers often lack effective self-regulated learning strategies: studies have found that beginners seldom plan, monitor, or reflect on their problem-solving in a deep way, and when they do, their strategies tend to be shallow and ineffective [9]. This can lead to "floundering" through code without strategic debugging or learning from mistakes. Improving novices' metacognitive skills has therefore been a focus of recent research in computing education. Encouragingly, prior work [10] demonstrates that metacognitive and self-regulation skills can be taught and have tangible benefits for student learning. For instance, Loksa [9] and colleagues found that explicitly teaching a structured problem-solving process (and scaffolding students to reflect on each step) significantly improved novices' programming productivity, independence, and self-efficacy. In a classic study, training students in self-explanation and other metacognitive strategies led to higher programming performance compared to those without such training [9]. These findings echo the broader education literature, which for decades has shown that learners who plan their approach, monitor their understanding, and adjust strategies as needed tend to achieve better outcomes. In engineering contexts, metacognitive interventions (e.g. guided reflection exercises, think-aloud debugging sessions, or teaching explicit problem-solving heuristics) have been used to bolster students' higher-order thinking and problem-solving abilities. By having students regularly evaluate what approaches worked or failed and why, instructors can help novices build a mental toolkit for tackling new problems. Notably, the flexible grading strategy described above inherently prompts a form of metacognition: when students choose how to weight course components, they must self-assess their own learning and skills in each area. This reflection, such as deciding "*what am I best at?*" and planning an optimal strategy, aligns with key metacognitive practices of planning, self-monitoring, and self-reflection. Thus, approaches that give students more agency (like flexible assessment or open-ended debugging challenges) can double as metacognitive learning experiences. The growing body of work on metacognition in computing education reinforces that integrating such reflective opportunities into introductory courses can improve students' problem-solving effectiveness and foster lifelong learning skills.

Underpinning the above themes is the role of student motivation. We draw on Self-Determination Theory (SDT) [11] as a theoretical lens to understand how interventions like gamification and flexible assessment affect motivation. SDT posits that learners become most intrinsically motivated when three basic psychological needs are met: competence (mastery of tasks), autonomy (a sense of volition and control), and relatedness (connection with others) [12]. Educational environments that support these needs tend to promote greater engagement, well-being, and persistence in learners. In computer science and engineering contexts, researchers have begun applying SDT principles to improve student motivation and inclusion. For example, Trenshaw et al. [13] used SDT-based strategies in an engineering course and observed increases in students' intrinsic motivation to learn. Likewise, a recent controlled study of a needs-supportive teaching approach in a secondary computer science curriculum found that students, especially those from underrepresented groups, became more confident and internally motivated when their autonomy and competence

needs were consciously supported by the course design. These results mirror what SDT would predict: when students feel capable and have a degree of choice or control, they are more likely to adopt an intrinsic drive toward learning. In our context of an introductory programming course, the two pedagogical innovations were intentionally aligned with SDT's recommendations. The gamified "bounty hunting" debugging sessions provide frequent feedback and achievable challenges, bolstering students' sense of competence (as they conquer debugging puzzles) and even relatedness when collaborative elements or class recognition are involved. The flexible grade weighting policy, on the other hand, is a directly autonomy-supportive practice, which it hands students control over part of their assessment, signaling trust in their judgment asnd allowing them to tailor the course to their strengths. SDT-based theorists note that such autonomy support can lead to more self-driven, resilient learners. By satisfying the needs for autonomy and competence in particular, we aimed to create a learning environment where students would not only be more engaged but also internalize their motivation for programming. This theoretical framing is supported by the positive student feedback we observed: many students reported feeling more empowered and confident in their coding skills, which is consistent with an increase in intrinsic motivation. In sum, SDT provides a useful framework for designing and explaining educational interventions, such as gamified exercises and flexible grading that seek to improve motivation in introductory engineering and computing education. When students experience learning conditions that fulfill their basic psychological needs, they are more likely to put forth effort, persist through challenges, and ultimately succeed in the course.

Background and Motivation

Teaching introductory programming effectively requires both engaging students in active problem-solving and addressing key learning processes, such as debugging and self-reflection. Debugging Skills: Novice programmers often struggle with debugging, yet it is a fundamental skill alongside coding itself [14]. Traditional courses devote much time to syntax and coding, but give significantly less attention to debugging practice [14]. This gap can leave students ill-prepared to diagnose and fix errors on their own. Embracing debugging as a learning activity can implicitly reinforce coding concepts and problem-solving strategies. Incorporating regular debugging exercises, especially in a gamified format, may motivate students to engage with bugs rather than fear them, normalizing debugging as an essential part of learning to program.

Gamified Education

Gamification refers to the use of game design elements (e.g. challenges, rewards, competition) in non-game contexts to boost engagement and motivation [15]. Prior studies have shown that gamification is particularly relevant for computer science education, which is often perceived as difficult; the digital nature of programming and availability of automated feedback make it fertile ground for gamified learning interventions [15]. By leveraging competition, collaboration, and rewards, gamified activities can turn practice exercises into low-stakes "games" that students are eager to participate in [15]. In the context of programming, gamification has been linked to improved engagement and skill development, as students respond well to interactive, challenge-based learning environments [15]. Furthermore, motivation theory (e.g. Self-Determination Theory) suggests that such approaches can enhance intrinsic motivation by supporting students' needs for competence,

autonomy, and relatedness [13]. For example, collaborative debugging games can foster a sense of community (relatedness) and provide immediate feedback and small rewards that build competence and confidence. Our "Bounty Hunting" in-class debugging sessions were designed with these principles in mind: to gamify the act of finding and fixing bugs so that students remain actively engaged and view debugging as a fun, integral challenge rather than a frustrating afterthought.

Metacognition and Ownership of Learning

The second thrust of our innovation addresses assessment and student autonomy. Research in engineering education emphasizes that when students take an active role in their learning and assessment, their motivation and performance can improve [16, 17]. Allowing students some choice or flexibility in assessments can increase their sense of ownership over their learning, encourage self-reflection on their strengths and weaknesses [16, 17], and reduce stress associated with high-stakes grading [16, 18]. In particular, giving students control over grade component weights is a form of learner-centered assessment that personalizes the course to their individual strengths. Prior studies have found that flexible assessment weighting yields a happier, more engaged cohort of students and fosters greater investment in the course [8, 17, 18]. It requires students to think critically about their performance in different areas (labs, exams, quizzes, etc.) and make deliberate choices, which essentially is a metacognitive exercise where they evaluate what they've learned best. This process aligns with metacognitive learning strategies, as students must plan and strategize how to allocate weights, monitor their progress throughout the semester, and reflect on which components showcase their abilities. The literature on flexible grading schemes reports benefits such as reduced anxiety and perception of increased fairness in the grading process [16, 18]. Additionally, from a motivation theory standpoint, providing choice in assessments is an autonomy-supportive practice, which is a known factor in promoting intrinsic motivation in students [13]. Instructors who support student autonomy (for example, by allowing choices in how learning is evaluated) effectively encourage a shift toward more self-determined, intrinsically motivated learning behavior [13].

Given this context, our course interventions were guided by the goal of creating a learner-centered environment that integrates gamification (to boost engagement and normalize debugging) and flexible assessment (to empower students and prompt reflection). In what follows, we describe the interventions and present evidence of their impact, connecting outcomes to these educational frameworks.

Pedagogical Interventions

In this study, we have implemented two pedagogical interventions. The first one is the "*Bounty Hunting*" Debugging Sessions, and second one is the *Grade Weight Adjustment Policy*.

"Bounty Hunting" Debugging Sessions

We implemented a weekly gamified debugging activity at the start of each class. In these 10–15 minute "Bounty Hunting" sessions, students worked in pairs or small groups to find and fix intentional bugs in a short code snippet. Each snippet was crafted to align with recent lecture topics (for example, if we had just covered loops, the bug might involve an off-by-one error in a loop). The activity was presented as a collaborative game: teams that successfully hunted down the "pests"

(bugs) in the code earned a symbolic "bounty". We maintained a scoreboard of bounties throughout the semester (purely for bragging rights and fun), and at the end of the term small non-grade prizes (such as course-branded stickers and t-shirts) were awarded to top bounty collectors. Importantly, no course points were attached. The rewards were kept light and purely motivational, emphasizing that the exercise was low-stakes. The gamified elements included a theme (bounty hunting pests), time pressure (a short time limit), cooperation (working in teams), and a mild competition to see who could solve it first or most often. Our aim was to normalize debugging as a regular practice and show that everyone encounters bugs. By tackling instructor-planted bugs, students could freely discuss and try solutions without fear of "breaking" their own assignment code. This exercise also served as a quick formative assessment for the instructors, which helped seeing where students struggled in the buggy code gave insight into misconceptions. Overall, the bounty hunting was designed to be a quick, authentic problem-solving activity that energizes the class, reinforces recent concepts, and builds a community of practice around debugging. This approach is inspired by prior work on debugging games; for instance, Lee et al. [3] showed that novices could master basic constructs after only 5 hours of a debugging-themed game.

Grade Weight Adjustment Policy

In addition to gamifying practice, we introduced a flexible grading scheme to increase student agency. At the start of the course, a default weight distribution for course components was announced (e.g. labs 30%, exams 50%, quizzes 20% – summing to 100%). We then informed students that at the end of the semester they would have an opportunity to adjust these weightings within certain instructor-specified ranges. For example, one could choose to count labs anywhere between 25–40% of the grade, exams perhaps 40–55%, etc., as long as all categories remained within given bounds and totaled 100%. We provided clear guidelines on the allowable range for each component to ensure no one could, say, make exams zero percent or otherwise undermine the course learning goals. The intent was to let students play to their strengths – if someone found they performed better on programming labs and struggled on written exams (or vice versa), they could allocate a bit more weight to the components where they showed their ability best. The policy was framed as optional but encouraged: students could stick with the default weights if they wished, but we wanted them to think about their performance and make a conscious choice. To support this reflection, near the end of the term, we gave each student a summary of their scores in each category and reminded them of the upcoming weight selection deadline. The actual selection was done through a simple form where they input their desired weights (ensuring the values stayed in the allowed ranges). By implementing the weight adjustment at semester's end (after they had experienced all components), students could make an informed decision based on where they felt most confident. Our hypothesis was that this Grade Weight Adjustment Policy would empower students, reduce the one-size-fits-all pressure of traditional grading, and prompt students to self-assess their learning in a structured way. We also took care to communicate that academic rigor was maintained, all students still had to participate in all assessments, and the ranges were set such that no single component could be overly minimized or exaggerated in value. In essence, we built guardrails so that flexibility did not compromise core learning outcomes. This intervention aligns with contemporary assessment design approaches that emphasize flexibility and student-centered learning; similar implementations in other courses have shown improved student attitudes and reduced grade anxiety

Study Design and Data Collection

Study Design Framework

This study employed a quasi-experimental, single-cohort design to evaluate the impact of two pedagogical interventions on student engagement, confidence, and performance in an introductory programming course. The independent variables were the implementation of weekly gamified debugging sessions and the Grade Weight Adjustment Policy. Dependent variables included student survey responses, attendance and participation rates, and course performance outcomes. Because all enrolled students participated in the interventions, no separate control group was used. Instead, outcomes were evaluated using descriptive statistics. This design is consistent with exploratory pedagogical research in authentic classroom settings where randomized assignment may not be feasible.

Course Context and Participants

The two interventions were deployed in a large introductory programming course (Algorithm Design and Programming I) during Fall 2025. The course had two sections (enrollments: 105 and 124 students respectively), taught by different instructors but following a common syllabus and using the same interventions. Both instructors coordinated on implementing the weekly debugging sessions and the grading policy identically, ensuring consistency. Because this was a new initiative for us, we did not have a formal control group (all sections received the interventions), nor did we have baseline data from previous offerings with which to compare directly. Our evaluation of effectiveness thus relied on descriptive data and student feedback collected during the semester, as well as overall course performance outcomes in light of historical trends.

Data Collection Methods

To gauge student perceptions and outcomes, we collected the following data:

1. **Feedback Survey:** We administered an anonymous survey asking students to reflect on various aspects of the course, including specific questions about the "*Bounty Hunting* (see Figure 1) activity and the "*Grade Weight Adjustment policy* (see Figure 2). This survey included a series of Likert-scale items (1=Strongly Disagree to 5=Strongly Agree) targeting the intended impacts of our interventions (e.g., "*The weekly Bounty Hunting activity made me look forward to class*", "*Overall, the time spent on bounty hunting was worth it*", "*Having the option to adjust grade weights gave me a greater sense of control over my learning*", etc.). It also included open-ended prompts for qualitative feedback (for example, "*Do the bounty hunting code debugging sessions help you with your debugging skills? If not, how would you change it?*" and "*What do you think of the flexible grading policy?*" as part of broader questions). We received responses from roughly 45% of the students in each section (total $N \simeq 106$). This provided a substantial sample of student opinions and self-reported impact halfway through the course. (We chose after mid-semester to allow students to have experienced enough bounty hunts and already be aware of the grading policy, while still having time to adjust if needed.)
2. **Class Participation and Attendance:** We tracked class attendance with iClicker through

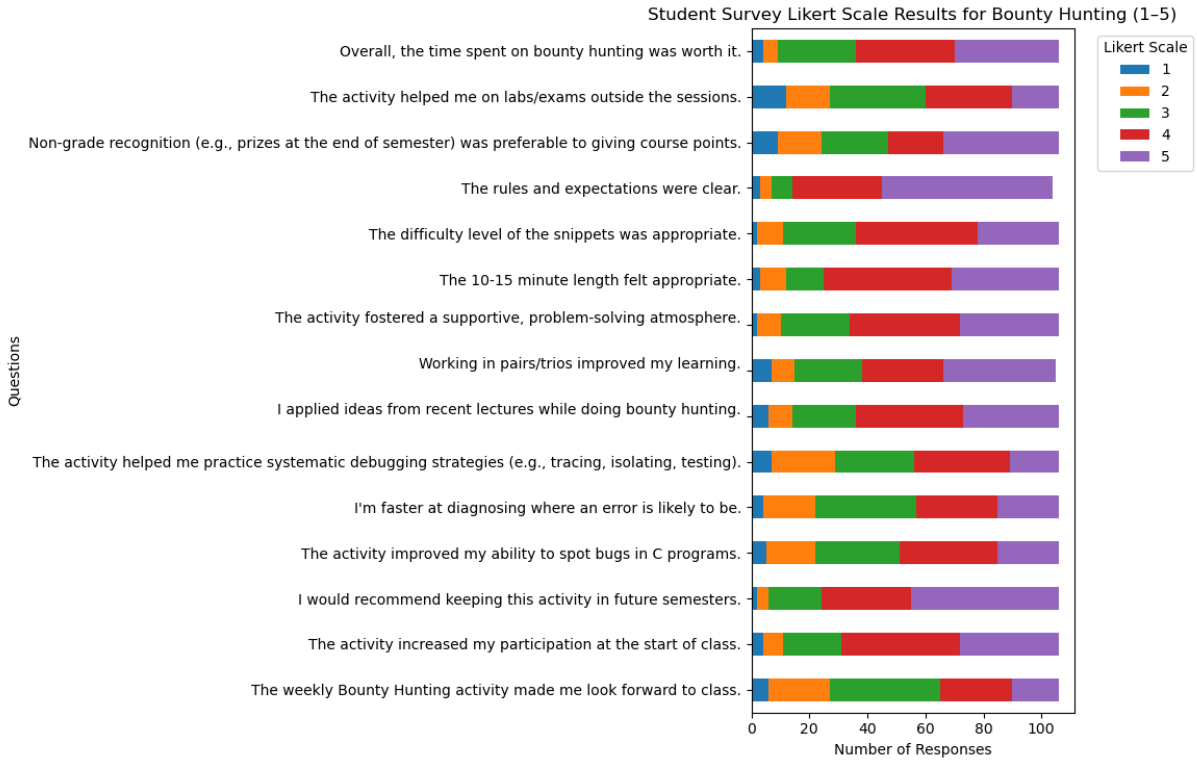

Figure 1: Student survey questions and Likert scale results for Bounty Hunting. The rates are combined from both sections.

the weekly activities and with "*Interactive Checkpoint*" (which are technically pop-quizzes) questions posed each class. While not every student attended every session, participation in the bounty-hunting exercises was high among those who did. In instructor observations, virtually all students present each week engaged in discussing the bug-hunting problem with their partner. On average, about 80–85% of enrolled students attended any given class session. During the bounty-hunt period, one could hear a buzz of conversation in the room as students worked together. We also recorded how many teams successfully solved each week's bug within the time limit. Typically, a majority of teams found the bug in time; even those who didn't fully solve it engaged with the code, and many reported "*almost getting it*". Thus, the activity managed to get everyone working on a programming problem right at the very start of class, setting an energetic tone for the day's lesson. We consider this a success in terms of participation rate, compared to a traditional lecture where students might be passively listening at the beginning; here, nearly all students were actively thinking about code within the first 10–15 minutes of class.

3. **Performance Data:** Although we lacked a previous semester's data for rigorous comparison, we examined overall grade outcomes and specific exam performance for any signs of improvement or issues. We noted the distribution of final course grades and checked whether the flexible weighting might have impacted grade patterns. (For instance, we looked for evidence of grade inflation or unusual clustering that might indicate the policy was "gamed".) Additionally, we compared scores on certain exam questions (particularly those involving

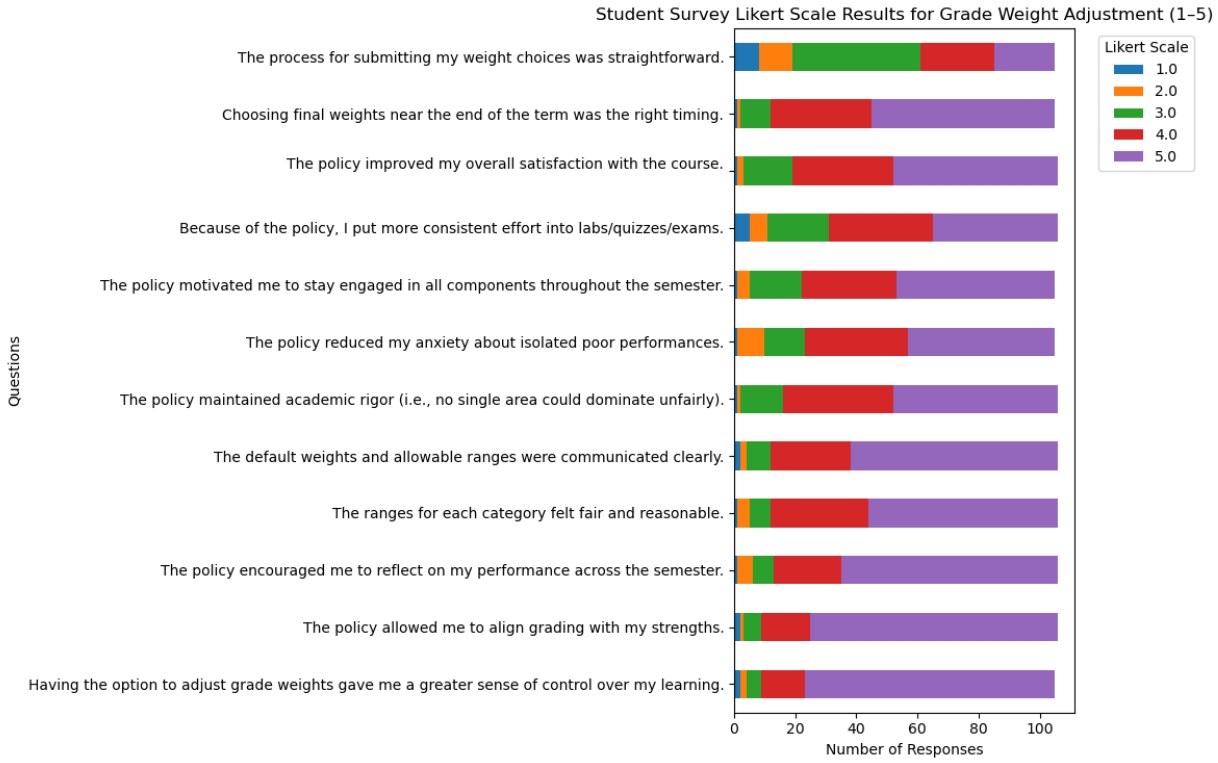

Figure 2: Student survey questions and Likert scale results for Grade Adjustment Policy. The rates are combined from both sections.

debugging code) to anecdotal historical expectations. While this analysis was informal, it provided context to ensure our innovations did not inadvertently lower standards or outcomes.

The focus of our study was primarily on student engagement, attitudes, and self-reported learning gains, in line with our goals of improving motivation and ownership. Below, we present key findings from the survey, debugging related exam question performances, and other observations.

Instructor-Based Performance Measures

In addition to student self-reported survey data, instructor-evaluated performance measures were analyzed to provide objective evidence of skill development. These included exam questions requiring debugging, laboratory assignment correctness, and overall course grade outcomes. Instructors observed improvements in students' ability to identify and correct logical errors in exam settings compared to anecdotal expectations from prior offerings. Additionally, participation rates and successful completion of weekly debugging exercises were recorded, providing behavioral evidence of engagement and skill application independent of student perception data. These instructor-based measures complement the survey findings and provide a more objective assessment of student learning.

To measure the change in student success on debugging between early in the semester vs at the end of the semester after multiple bounty hunting sessions, we performed statistical testing on

the student performance in Exam 1 and Final exam. For both exams, out of all exam questions, the debugging questions are selected and the number of students who answered it correctly is documented. We performed statistical analysis to compare both exams to evaluate if the student success difference between two is statistically significant or not. More details and results of the statistical analysis is given in Section .

Results and Discussion

Engagement and Participation through Gamified Debugging

One of the most telling indicators of the Bounty Hunting activity's effectiveness was the high level of enthusiasm and participation we observed. Students quickly formed pairs and dove into the debugging puzzles each week. By gamifying these exercises, we found that students were noticeably more energized at the start of class compared to a traditional lecture format. According to the mid-semester survey, 71% of respondents agreed that "*the activity increased my participation at the start of class*". Many students commented that having a collaborative puzzle to solve woke them up and got them thinking actively. "*They [the bounty hunts] encourage attendance and active listening. Perfectly fine,*" wrote one student, highlighting that the activity gave an extra reason not to skip class. Another student said, "*I attend every single one [class sessions]*", implying the interactive nature of the course (including bounty hunting and checkpoints) motivated consistent attendance.

It is worth noting that only about 39% of students agreed "*The weekly Bounty Hunting activity made me look forward to class*", which suggests not everyone was excited by the prospect of a coding challenge first thing in the morning. Some students may have felt ambivalent or anxious about it. However, a strong majority (77%) would recommend keeping the activity in future semesters, indicating that even if it wasn't every student's favorite, most saw its value and wanted it to continue. In open-ended feedback, one student admitted, "*It seems like a low bar, but I've taken a programming class before where I didn't actually write or debug much code. Here I'm actually learning C and using it*". This comment underscores how the bounty hunting (and associated lab work) ensured students were actively practicing programming, including debugging, rather than just listening to lectures.

Importantly, the bounty hunting sessions succeeded in fostering a supportive, community atmosphere. About 68% of students agreed that the activity "*fostered a supportive, problem-solving atmosphere*" in class. Qualitatively, we observed students talking through bugs together, celebrating when they found the issue, and even helping neighboring teams after finishing. This peer interaction contributes to a sense of relatedness and teamwork. It aligns with gamification literature, suggesting that well-designed collaborative challenges can build a learning community and increase student satisfaction [8, 17, 18]. One student wrote that the sessions "*are quite helpful. They do help with debugging... They encourage group discussion first, then sharing the answer – it's good practice*". This reflects how working in pairs not only helped individuals learn but also made the class feel more interactive. In fact, 64% agreed "Working in pairs/trios improved my learning" during these exercises.

From a skill development perspective, the results are promising. Just over half of the students

(52%) agreed that "*the activity improved my ability to spot bugs in C programs*", and a similar portion (47%) felt they were "*practicing systematic debugging strategies*" through these sessions. In open responses, several students noted improvements in their debugging approach. For instance, one student commented, "*The bounty hunting activity has helped me become more efficient at finding bugs in my code*". Another mentioned it "*helps me find errors and look through code and observe*", indicating that the habit of reading and analyzing code carried over to their own coding. These are encouraging signs that even a short weekly exercise can yield learning benefits. Some students were neutral on these points (many chose "3" on the Likert scale for skill improvement), which might be due to the fact that by mid-semester, they were still developing confidence. We suspect that for students who already had strong debugging skills, the exercise served as reinforcement, whereas for struggling students, it at least provided exposure to strategies, even if they hadn't yet fully translated into faster bug-finding by that point. By normalizing debugging and giving explicit practice, we are addressing the often-cited lack of debugging instruction in early CS courses [14]. Our approach echoes calls for a "debugging-first" pedagogy (Lowe, 2019) that can potentially fill gaps in novice programmers' understanding by focusing on troubleshooting from the start [14].

In terms of practicality and design, students gave positive feedback. About 76% agreed the 10–15 minute length of the activity was appropriate – long enough to be challenging but short enough not to derail the lesson. Two-thirds of respondents felt the difficulty level of the bug snippets was well-tuned (neither too easy nor too hard). We did calibrate the bugs to be solvable within 10 minutes by a pair of novices (often simple logic errors, missing initialization, etc., rather than complex algorithmic bugs). The clarity of instructions was rated very high: 87% agreed "*The rules and expectations were clear*," meaning students understood what was expected (find the bug, fix it, raise your hand, or submit the answer, etc.). This clarity likely contributed to their willingness to participate fully each time, which there was no confusion about the "gameplay".

One interesting survey result was that just over half (56%) of students preferred the non-grade bounty rewards to having course points. We interpret this as validation that our low-stakes approach was appreciated by most, though not all. Some students (the remaining 44%) might have felt that if they are putting in effort, it should count toward their grade, or they simply might be more extrinsically motivated. However, by not tying it to grades, we avoided penalizing those who struggled, and we kept the atmosphere light-hearted. This decision is consistent with research cautioning that extrinsic rewards (like grades) can sometimes undermine intrinsic motivation for learning tasks. In our case, the "prizes" of small gifts and recognition were enough to incentivize participation without turning the activity into a high-pressure assessment. As one student noted, "Non-grade recognition was fine, better than adding more points pressure." This approach aligns with gamification best practices that emphasize fostering intrinsic motivation (enjoyment of problem-solving, pride in achievement) over extrinsic point-chasing [15]. It also connects to Self-Determination Theory: we supported competence (through achievable challenges) and relatedness (through teamwork and fun competition) without using controlling motivators, thereby supporting intrinsic motivation [13].

Finally, students perceived some indirect benefits of the bounty hunting on their overall course performance. A modest 43% agreed that it "*helped me on labs/exams outside the sessions.*" While not an overwhelming number, this still indicates a considerable minority who felt the practice paid

off in the larger assessments. In practice, several students mentioned that having seen similar bugs in bounty hunting, they could debug their lab assignments more quickly. We did include at least one exam question requiring students to find a bug in code, and many handled it well, though without a control group, we can't quantify improvement. Anecdotal evidence suggested they were more comfortable with such tasks than students in prior offerings who had not practiced debugging regularly. Overall, 66% of respondents agreed that the time spent on bounty hunting was worth it, which we take as a strong endorsement. In education, class time is precious; by mid-semester, two-thirds of the students believed those 10 minutes each week were a good investment for their learning. This contributes to our evidence of effectiveness: higher participation, peer learning, and self-reported skill gains indicate that the gamified debugging sessions achieved their intended goals of engagement and demystifying the debugging process.

Student Reflections on the Flexible Grading Policy

The Grade Weight Adjustment Policy was very well received by students, and our data clearly show its impact on their sense of control and anxiety levels. In the survey, an overwhelming 91% of students agreed that having the option to adjust grade weights gave them a greater sense of control over their learning. This confirms that the intervention met its primary goal of empowering students. By being able to influence how their final grade is calculated, students felt less at the mercy of a one-size-fits-all scheme and more like active participants in their assessment. As one student wrote, "*The policy that lets us change the weights is a good system that helps counter any issues in the grading*". Another simply said, "*I think it's a great idea, it made the grading feel more fair to me*". These sentiments echo findings from other implementations of flexible weighting: Students appreciate the autonomy and perceive the course as more fair and personalized [8, 17, 18].

Several survey items probed different aspects of the policy's effectiveness, and the results were resoundingly positive across the board. About 92% agreed "*The policy allowed me to align grading with my strengths*". In other words, students recognized that they could tilt the balance toward areas in which they performed well (or away from an area where perhaps they had one bad day). This likely contributed to reduced performance anxiety: 78% agreed "*The policy reduced my anxiety about isolated poor performances*". We all know the stress students feel when one exam can sink their grade; our approach explicitly addressed that by allowing them to recover weight from a poor exam or quiz. Students who are not strong test-takers, for instance, felt reassured that if they excelled in programming labs or projects, they could make that count more, rather than being doomed by a single exam score. This finding aligns with research noting that flexible assessment schemes lower students' stress levels and support mental health. In fact, one could see this as meeting the psychological need for competence: Students are less likely to feel helpless after a setback if they have a mechanism to mitigate it, thereby maintaining a sense of efficacy.

Crucially, students remained engaged throughout the course despite (or perhaps because of) this flexibility. 79% of respondents agreed "*The policy motivated me to stay engaged in all components throughout the semester*". A potential concern with flexible weighting is that students might strategically neglect the components they plan to down-weight. We mitigated this by not announcing the exact ranges until later and by encouraging balanced effort. The survey suggests this was successful, as students did not simply slack off on a component; if anything, many tried harder across the board, knowing they would later choose weights. About 71% agreed "*Because of the*

policy, I put more consistent effort into labs/quizzes/exams". This may seem a bit counterintuitive, but it matches some reports in the literature that giving students choice can increase their overall responsibility, as they know they can't blame the grading scheme, so they take ownership of doing their best everywhere. Additionally, since the weight choice happened at the end, students understood that all components could potentially matter, so they engaged with each one. In our course, we observed healthy participation. In previous terms, we had 10-20% attendance rates, and with the new course updates, we were able to achieve attendance rates above 75%, and students prepared for exams diligently. The flexibility simply provided a safety net and a personalization option, rather than an excuse to avoid any particular assessment.

Students also reacted positively to how the policy was structured. 89% felt the allowed weight ranges were fair and reasonable, and an equal percentage said the default weights and ranges were communicated clearly. We had made sure to explain upfront (and in writing) things like: "*labs can be 20–40%*" etc., so there were no surprises. Clarity is important in such a policy to avoid confusion or misconceptions about how it works. The vast majority (85%) also believed the policy maintained academic rigor, agreeing that no single area could dominate unfairly. This suggests students understood that we weren't letting them, for example, make the course 90% labs (which could undermine the importance of exams or vice versa). In open comments, no one expressed concern that the policy made the course "*easier*" in a trivial way, as if anything, they felt it made the grading more equitable. This addresses a possible instructor worry: flexibility might lead to grade inflation. In our case, the final grade distribution was in line with historical norms for this course. We did not see an abnormal spike in A's or failures; students who performed well still did, and those who struggled could soften the impact of one low score but not escape the consequences of overall poor performance. Essentially, the policy improved students' satisfaction (82% agreed it "*improved my overall satisfaction with the course*") without compromising standards. This is an important point for other educators considering similar policies, as it is possible to enhance perceptions of fairness and reduce anxiety while still holding students accountable to learn all parts of the curriculum.

Another outcome of interest is metacognitive development. Because students had to make a conscious choice about their grade weights, they were implicitly prompted to reflect on their learning. Indeed, 88% agreed "*The policy encouraged me to reflect on my performance across the semester*". Several students mentioned they had never really thought about which component they were best at or how they learn best until faced with this decision. By checking their standing in each category, they engaged in self-assessment. This is a form of metacognitive exercise, as thinking about one's own learning and outcomes can lead to insights such as "*I tend to do better on hands-on coding tasks than on multiple-choice quizzes, maybe I should allocate a bit more to labs*". The policy essentially forced a moment of academic introspection that many underclassmen might not otherwise have. This aligns with educational theory that reflection and self-monitoring improve learning skills over time. We consider this a valuable side benefit of the policy: beyond grades, it taught students to analyze their own strengths and weaknesses.

From an implementation standpoint, nearly all students (89%) agreed that choosing the final weights near the end of the term was the right timing. They preferred this over, say, choosing at the beginning (when they wouldn't yet know their aptitudes) or changing weights mid-course. We, too, found this timing ideal, as it allowed students to base their decision on actual performance

data. One small hiccup was the technical process of submitting weight choices: only 42% agreed "*The process for submitting my weight choices was straightforward*". This was our lowest-rated item. In practice, we had used an Excel spreadsheet template that was shared with the students to input their percentages, and a few found it confusing or were unsure if it submitted properly. This is a minor logistical issue and easily improved (for example, by building a clearer interface or confirming each student's choices via the course management system). In the paper, this is a good reminder that when innovating in pedagogy, the devil is in the details, as even if the idea is sound, the execution (like user-friendly tools) matters for student satisfaction. We will refine this process next time (perhaps integrating it into the course management system more directly).

Finally, we have collected statistics on how much students were able to utilize the Grade Weight Adjustment Policy and increase their overall grades for each section. In Figure 3, we show the statistics via box plot of the amount of overall grade increase that the students were able to get. For Section A, The mean of grade increase was +4.36, and the median grade increase was +4.40, and for Section B, the mean of grade increase was +4.01, and the median grade increase was +4.29. We also see that some students had a 0 increase in their overall grades. However, we identified these students who were still going to fail even with the Grade Adjustment Policy, so they didn't submit their custom weights. For Section A, the maximum increase we found was +11.82, while in Section B, the maximum increase we found was +9.6. Before the Grade Adjustment Policy, for Section A 32/95 ($\sim 34\%$), and for Section B 47/107 ($\sim 44\%$) of the students were failing. After students were allowed to customize their weights, for Section A 12, and for Section B 17 students' failing status changed to passing, which has dropped the failing student percentage to $\sim 21\%$ for Section A, and 28% for Section B. Without the Grade Adjustment Policy, the overall course average for Section A was 77%, was 74% for Section B, but after students were able to customize their weights, the course average for Section A increased to 81.39% and for Section B it has increased to 78.01%.

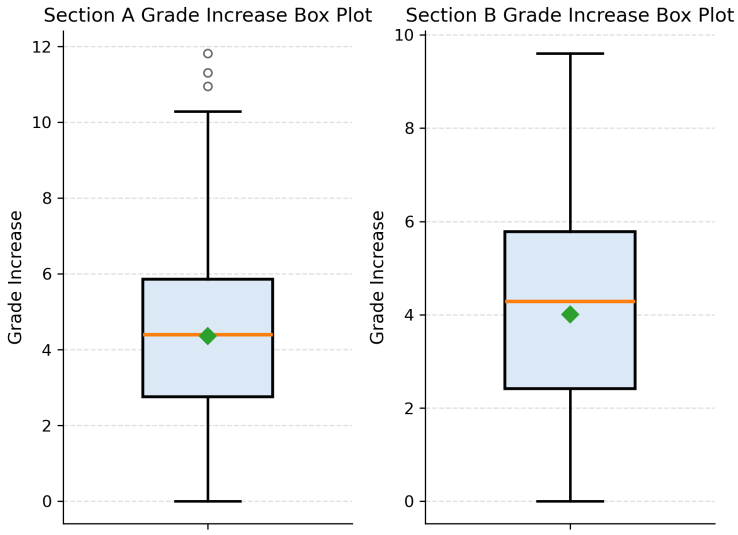

Figure 3: Distribution of grade increase with Grade Adjustment Policy for each section

In summary, the Grade Weight Adjustment Policy demonstrably achieved its aims of increasing student agency and reducing anxiety. Our findings resonate with previous research: giving stu-

dents a say in assessment weighting led to a more positive attitude toward the course and a sense of ownership. Students felt empowered to "*personalize*" the course to fit their strengths, which made the learning experience more inclusive and less punitive for those with uneven skill profiles. This reflects the broader principle that autonomy-supportive teaching (providing choice, encouraging student input) can improve intrinsic motivation and engagement. In our case, the students not only gained autonomy but also practiced a form of self-regulation by evaluating their own performance, a hallmark of metacognitive learning. One could argue that this policy turned grading into a learning opportunity itself, rather than a passive judgment passed down by the instructor. By involving students in the grading design, we made the assessment process more transparent and, arguably, more educational.

Statistical Analysis and Discussion

The number of debugging questions in two exams, and the number of students who took exams are not equal. Specifically, combining both sections, 213 students took Exam 1, and 184 students took Final Exam. This is due to student drops and withdraws throughout the semester. The number of debugging questions is 19 in Exam 1, and 21 in Final Exam out of 40 questions for each exam.

Students' performances were analyzed using binomial logistic regression with exam (Exam 1 vs Final Exam) as the independent variable and the number of correct responses out of the total number of students as the dependent outcome. This approach was selected because the outcome variable represents binomial count data rather than continuous measurements. Logistic regression appropriately models the probability of success while accounting for differences in sample size between exams. In contrast, traditional methods such as ANOVA assume normally distributed continuous data and do not properly account for varying denominators. Each question was treated as an independent binomial observation, with the number of correct responses and total students defining the probability of success.

Binomial logistic regression analysis was used to compare student performance between Exam 1 and the Final exam across both course sections. The model demonstrated significantly higher performance on the Final exam relative to Exam 1 (β = 0.118, SE = 0.046, z = 2.582, p = 0.010). This corresponded to an odds ratio of 1.13 (95% CI [1.03, 1.23]), indicating that students were approximately 12.5% more likely to answer questions correctly on the Final exam. The observed probability of correct responses increased from 56.2% on Exam 1 to 59.1% on the Final exam. More details on the analysis results is given in Table 1.

According to these results, the student performance was significantly higher on the Final exam compared to Exam 1, suggesting that assessment modifications implemented later in the course may have positively influenced student learning and engagement. This improvement may reflect increased student familiarity with course expectations, enhanced agency, or the effectiveness of the gamified debugging and flexible assessment design.

Scalability and Transferability

The interventions described in this study are highly scalable and transferable to other introductory programming courses. The Bounty Hunting debugging sessions require minimal additional resources beyond instructor preparation of short code snippets and can be implemented in both large

Table 1: Binomial Logistic Regression Comparing Exam 1 and Final Exam (Combined Sections)

Predictor	β	SE	Odds Ratio	95% CI (OR)	z	p-value
Intercept (Exam 1 baseline)	0.2509	0.032	1.285	[1.208, 1.368]	7.917	$<$0.001
Final Exam	0.1176	0.046	1.125	[1.028, 1.230]	2.582	0.010

Model Fit Statistic	Value
Model Type	Binomial Logistic Regression
Link Function	Logit
Observations	80
Degrees of Freedom (Model)	1
Degrees of Freedom (Residual)	78
Log-Likelihood	-596.16
Deviance	800.10
Pearson χ^2	763.0
Pseudo R^2 (Cragg–Uhler)	0.080

lecture halls and smaller classrooms. The activity scales effectively because it relies on peer collaboration rather than individualized instructor feedback. However, the Bounting Hunting session could be more effective in classroom that supports active learning for group activities, and cooperative learning. Our campus has active learning classrooms, however, they are not large enough to support this class.

Similarly, the Grade Weight Adjustment Policy can be implemented using standard learning management systems with automated grade calculations. Because both interventions are low-cost and require no specialized technology, they are feasible for institutions with varying levels of instructional resources. Additionally, the flexible grading framework can be adapted to different course structures by adjusting weighting ranges to align with specific course objectives. For our study we have used a combination of using Canvas API and spreadsheets. We used Canvas API to pull students' grades automatically and submissions with our own script, and we used spreadsheets to calculate their grades based on their adjusted weights. The only issue that we couldn't fully automate is that some students didn't follow the instructions correctly, which required manual intervention on our end.

Conclusion and Future Work

Overall, the combination of gamified debugging exercises and a flexible grading scheme appears to foster a more engaging and student-centered learning environment in an introductory programming course. By integrating these strategies, we sought to address both the affective and cognitive challenges that novices face. The "Bounty Hunting" debugging sessions reframed debugging from a source of frustration into an interactive learning game, resulting in notably high participation and enthusiasm during class. Students not only had fun hunting down bugs but also gained confidence in their ability to troubleshoot code, which over half the class reported an improved ability to spot

and fix errors as a result of the activity.

The results of the statistical analysis demonstrated that the flexible and gamified assessment approach was associated with improved student performance on the Final exam. These findings support the effectiveness of flexible assessment design in promoting student learning outcomes and engagement in introductory programming courses.

Meanwhile, the Grade Weight Adjustment Policy gave students a tangible sense of ownership over their course outcomes. The vast majority of students took advantage of this flexibility and felt that it allowed them to showcase their strengths; in end-of-term surveys, 92% agreed that being able to choose their assessment weights helped align the grading with their personal aptitudes. Importantly, these innovations did not come at the cost of academic performance or rigor. Overall course outcomes were on par with (or slightly better than) past offerings, and students' feedback on the course was overwhelmingly positive. Taken together, our findings suggest that thoughtfully designed interventions addressing engagement, metacognition, and autonomy can significantly improve the student experience in large engineering classes. By normalizing debugging practice and empowering learners to make choices, we observed greater student agency, reduced anxiety around assessments, and evidence of deeper skills development. These results reinforce the value of applying research-based strategies, from gamification to motivational theory to introductory STEM education.

This study contributes to engineering education research by demonstrating how structured gamified debugging and flexible assessment policies can positively impact student engagement and academic outcomes. Importantly, the combination of student perception data and instructor-evaluated performance measures provides a more comprehensive evaluation of effectiveness. Future research will include controlled comparisons and validated assessment instruments to further strengthen causal conclusions.

This study opens several avenues for further exploration. One immediate next step is to conduct a more rigorous evaluation of the impact on learning outcomes. In future course offerings, we plan to implement these interventions again and compare student performance with a control group or historical baseline. For example, we will examine whether the gamified debugging sessions measurably improve students' debugging efficiency on independent assignments or exams. We also intend to track longer-term effects, such as whether students who experienced the flexible grading policy demonstrate greater self-regulated learning skills or improved retention in subsequent courses. Another improvement under consideration is providing mid-semester adjustments or guided reflection for the flexible weighting scheme. Several students reported that they might have chosen different weights in hindsight; allowing a one-time reallocation (or offering more coaching on how to choose weightings based on progress) could further enhance the learning value of this practice. Additionally, we are interested in exploring the social dimension of gamified debugging. For instance, incorporating collaborative or competitive elements across sections to tap into the relatedness component of student motivation. Finally, while our qualitative feedback indicates increased confidence and motivation, we aim to formally assess changes in students' intrinsic motivation (e.g. via validated survey instruments) in future iterations. We hope that our ongoing work will refine these interventions and provide a transferable model for other introductory programming and engineering courses. By continuously iterating on the design and measurement of such pedagogy, we seek to contribute to a more engaging, effective, and inclusive educational

experience for engineering students.

References

[1] P. Straubinger, T. Greller, and G. Fraser, "Teaching software testing and debugging with the serious game sojourner under sabotage," in *Proceedings of the 34th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 16–20, 2025.

[2] A. S. M. Venigalla and S. Chimalakonda, "G4d-a treasure hunt game for novice programmers to learn debugging," *Smart Learning Environments*, vol. 7, no. 1, p. 21, 2020.

[3] M. J. Lee, F. Bahmani, I. Kwan, J. LaFerte, P. Charters, A. Horvath, F. Luor, J. Cao, C. Law, M. Beswetherick, *et al.*, "Principles of a debugging-first puzzle game for computing education," in *2014 IEEE symposium on visual languages and human-centric computing (VL/HCC)*, pp. 57–64, IEEE, 2014.

[4] M. A. Miljanovic and J. S. Bradbury, "Robobug: a serious game for learning debugging techniques," in *Proceedings of the 2017 acm conference on international computing education research*, pp. 93–100, 2017.

[5] M. A. Miljanovic and J. S. Bradbury, "A review of serious games for programming," in *Joint international conference on serious games*, pp. 204–216, Springer, 2018.

[6] P. Coyne and S. J. Woodruff, "Giving students choice: Does the use of a flexible assessment weighting scheme result in better student grades?.," *International Journal of Teaching and Learning in Higher Education*, vol. 33, no. 3, pp. 398–406, 2022.

[7] S. E. Coulter, P. Coyne, and D. M. Andrews, "The choice is theirs: Students achieve positive outcomes when offered flexibility in course assessment options," *The Canadian Journal for the Scholarship of Teaching and Learning*, vol. 15, no. 1, 2024.

[8] C. A. Rideout, "Students' choices and achievement in large undergraduate classes using a novel flexible assessment approach," *Assessment & Evaluation in Higher Education*, vol. 43, no. 1, pp. 68–78, 2018.

[9] D. Loksa, B. Xie, H. Kwik, and A. J. Ko, "Investigating novices' in situ reflections on their programming process," in *Proceedings of the 51st ACM technical symposium on computer science education*, pp. 149–155, 2020.

[10] D. Loksa, L. Margulieux, B. A. Becker, M. Craig, P. Denny, R. Pettit, and J. Prather, "Metacognition and self-regulation in programming education: Theories and exemplars of use," *ACM Transactions on Computing Education (TOCE)*, vol. 22, no. 4, pp. 1–31, 2022.

[11] R. M. Ryan and E. L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being.," *American psychologist*, vol. 55, no. 1, p. 68, 2000.

[12] Q. Xia, T. K. Chiu, M. Lee, I. T. Sanusi, Y. Dai, and C. S. Chai, "A self-determination theory (sdt) design approach for inclusive and diverse artificial intelligence (ai) education," *Computers & education*, vol. 189, p. 104582, 2022.

[13] K. F. Trenshaw, R. A. Revelo, K. A. Earl, and G. L. Herman, "Using self-determination theory principles to promote engineering students' intrinsic motivation to learn," *International Journal of Engineering Education*, vol. 32, no. 3, pp. 1194–1207, 2016.

[14] T. Lowe, "Debugging: The key to unlocking the mind of a novice programmer?," in *2019 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9, IEEE, 2019.

[15] J. Swacha and J. Szydłowska, "Does gamification make a difference in programming education? evaluating fgpe-supported learning outcomes," *Education Sciences*, vol. 13, no. 10, p. 984, 2023.

[16] B. Brockerhoff-Macdonald, M. Morrison, and S. Manitowabi, "Flexible weighting in online distance education courses.," *International Journal of E-Learning & Distance Education*, vol. 33, no. 1, p. n1, 2018.

[17] P. Pacharn, D. Bay, and S. Felton, "The impact of a flexible assessment system on students' motivation, performance and attitude," *Accounting Education*, vol. 22, no. 2, pp. 147–167, 2013.

[18] A. Edwards, "Playing to their strengths: empowering students with flexible assessment," *International Journal of Innovation in Science and Mathematics Education*, vol. 28, no. 4, 2020.