

Teaching OSS Contribution Through a GitHub Bot

Misan P. Etchie¹, Karissa Smallwood¹, Igor Steinmacher¹, and Marco Gerosa¹

Northern Arizona University, Flagstaff, AZ, USA
{mpe45,kns472,igor.steinmacher, marco.gerosa}@nau.edu

Abstract. Software developers seeking to contribute to open source software communities face barriers that hinder onboarding, including navigating unfamiliar contribution workflows and issue-tracking conventions and obtaining timely and actionable guidance during their first interactions with a project. Although platforms like GitHub Classroom have addressed assignment distribution infrastructure, they fall short in providing interactive, context-sensitive support within the developer workflow itself. In this paper, we present the development and evaluation of the OSS-Doorway GitHub bot, which monitors student interactions, triggers automated LLM-based validation, and delivers instant feedback directly within GitHub issues and pull requests. OSS-Doorway structures learning through a gamified quest progression system, in which students unlock sequential challenges, earn experience points, and track advancement through a leaderboard. We deployed OSS-Doorway in a mixed undergraduate and graduate open source software development course and evaluated it in a post-course survey ($N = 21$). Students reported that the quest-based workflow was generally more engaging than traditional assignments and useful for reinforcing lecture concepts, and they appreciated the immediacy of automated feedback. At the same time, overall recommendation was only moderate, with most respondents classified as passives, and students identified the validator’s reliability and feedback specificity as recurring frustrations. Taken together, our findings suggest that OSS-Doorway is a promising but not yet fully polished onboarding tool and point to concrete directions for improving this type of bot.

Keywords: Open Source Software · Software Engineering · Automated Feedback · Gamification · Large Language Models · OSS Onboarding

1 Introduction

The importance of Open Source Software (OSS) and its accompanying collaborative communities in our current technological ecosystem cannot be overstated. Foundational technologies like Linux underpin much of modern computing by providing the stable, secure base layer many organizations rely on. Computer systems are backed by a massive ecosystem of open libraries available in repositories such as GitHub, which has served as the go-to platform for collaborative software development. Because of this, many courses now rely on GitHub-based

workflows to expose students to version control, pull-request based code review, issue tracking, and collaborative development conventions [11, 14]. Despite these efforts, students struggle to navigate GitHub effectively in educational settings [11].

Newcomers to OSS often struggle with contribution-process hurdles such as locating relevant documentation, understanding repository structure, and following project conventions, and tools such as FLOSScoach were designed to reduce these entry barriers [31]. Empirical evidence highlights barriers during first contributions that involve communication and interaction with the community, including challenges in getting timely, helpful responses [30]. These challenges manifest as frustration, shallow engagement, and limited skill transfer [31], often leading newcomers to give up on contributing to OSS projects.

To address these challenges, we introduce OSS-Doorway, a gamified system that structures OSS coursework as a sequence of progressive quests and tasks delivered entirely through GitHub. In our system, GitHub issues are transformed into tasks within quests, where students interact with coursework that pertains to GitHub workflows, including creating documentation, running GitHub Actions, validating pull requests, and making code reviews. Tasks are embedded in a broader quest structure that supports incremental onboarding and pacing. Alongside this, an integrated automated validator provides iterative, timely feedback on student submissions, enabling iterative refinement, personalized support, and reducing reliance on delayed grading, which prior work suggests is less effective [19].

OSS-Doorway is built on four principles. First, the system aims to minimize context switching by keeping students within GitHub for task discovery, submission, and feedback. Second, it aims to provide timely formative feedback to support learning during task completion rather than after submission deadlines. Third, the system supports progressive onboarding by organizing coursework into manageable, dependency-aware tasks that reflect real-world software engineering workflows. Finally, OSS-Doorway enables instructors to author, review, and deploy coursework safely through a draft-to-publish pipeline, supporting iterative course design without exposing students to incomplete content.

We deployed OSS-Doorway in a mixed undergraduate/graduate open source course and conducted an empirical evaluation using a post-course survey. Students generally reported that the quest-based workflow was more engaging than traditional assignments and helpful for reinforcing lecture concepts, improving familiarity with GitHub, and clarifying how to interact with instructors and teaching assistants through GitHub. However, overall recommendation was only moderate (NPS = +4.8, with most respondents classified as passives), and students identified clear pain points around validator reliability and onboarding clarity. Our results provide tentative support for the idea that gamified task structuring can support learning and onboarding in software engineering education, while highlighting concrete limitations that future work needs to address.

This paper makes the following contributions:

- We present the design and implementation of *OSS-Doorway*, an end-to-end quest system for coursework that facilitates mechanisms for task sequencing, progression gating, automated validation, and formative feedback through GitHub.
- We report empirical findings from a classroom deployment, identifying strengths in engagement and onboarding as well as limitations in automated feedback clarity and artifact support.
- We discuss design insights and takeaways for feedback mechanisms in gamified GitHub-based software engineering education.

2 Background and Related Work

Our system sits at the intersection of two challenges in OSS-centered education: (i) newcomers often struggle to navigate workflows and identify next steps in repositories, and (ii) feedback needs to be timely and actionable during task completion. We therefore review work on onboarding and scaffolding in OSS/GitHub contexts, approaches that use gamified or task-based structures to sustain engagement, automated assessment techniques for rapid formative feedback, and recent findings on how learners use LLM tools when instructions or feedback are difficult to interpret.

2.1 Onboarding and Newcomer Support in OSS and GitHub Workflows

To sustain OSS projects and to enable effective learning in software engineering education, onboarding new contributors is critical. Despite the importance of attracting and retaining newcomers, they often face steep learning curves due to technical and socio-cultural barriers [25]. A systematic review by Steinmacher et al. [32] organized 15 onboarding barriers into five broad categories: social interaction issues, gaps in newcomers’ prior knowledge, difficulty finding a way to start, documentation problems, and code-related hurdles. Across the surveyed studies, barriers related to social interaction and responsiveness suggest that social dynamics are central to whether newcomers persist beyond their first attempts [32]. Balali et al. [2] also found that 66

In addition to human mentors, OSS projects increasingly rely on workflow and tooling support that reduces the friction of making a first contribution. A widely adopted practice is to label curated beginner-friendly tasks, such as good first issue,” so newcomers can identify a feasible starting point. However, empirical work shows that these labels are not automatically sufficient: He et al. found that many good first issues” still include inadequate information for onboarding and proposed a recommendation approach that ranks issues by newcomer suitability rather than relying on labels alone [38]. Building on this direction, He et al. introduced GFI-Bot, an automated recommender that suggests personalized good-first-issues to newcomers on GitHub [13].

Automated validation and feedback embedded into contribution workflows serve as a second pillar of tooling support. In practice, pull-request based development is frequently paired with automated checks that provide immediate, objective signals about correctness and conformance, reducing repetitive maintainer effort and giving newcomers faster iteration cycles. The literature also emphasizes that immediate, automated feedback can shape how learners iterate and engage with tasks [19], and systematic reviews show that automated feedback generation is most dependable when paired with clear criteria and structured checks [16]. Complementing workflow-level automation, onboarding-specific tools can also reduce entry barriers. For example, Steinmacher et al. designed FLOSScoach, a portal that consolidates project knowledge and step-by-step contribution guidance, helping newcomers orient themselves and follow project conventions [31]. Alongside tools, practitioner-oriented guidance has also emerged to operationalize onboarding best practices through clear processes, discoverable tasks, and responsive communication norms [27].

2.2 Gamification in SE Education

Gamification, the use of game design elements in non-game contexts, typically to influence motivation, engagement, and persistence, has been the subject of study [7]. In educational settings, the core claim is not that points or badges “cause learning,” or necessarily the primary factor or driver of learning, but that appropriately chosen mechanics can structure participation, reward progress, increase attendance, boost user retention and interest, and sustain effort on difficult tasks [8]. Across higher-education contexts, evidence is mixed, but studies report gains in engagement and perceived learning when game elements are integrated to support course goals (e.g., structured progression via missions/quests/levels and timely feedback), rather than added as standalone superficial reward layers (e.g., points/badges/leaderboards) [34].

To design gamified educational systems, several works propose taxonomies and frameworks that enumerate reusable elements (levels, quests, leaderboards, badges, and challenges) and describe how they map to learning or motivational outcomes [8]. This line of work is especially relevant in software engineering education, where multi-step skills (tooling, collaboration, testing, and process) often benefit from staged practice and clear intermediate milestones. Gamification has been applied to domains such as documentation, bug reporting, testing, and process adherence, often with the goal of increasing participation and consistency in behaviors that students otherwise avoid [26]. More recent work continues to explore gamified structures to encourage repeated practice, deliberate feedback cycles, and sustained engagement in technically authentic environments (including tool-mediated settings) [33].

For OSS-Doorway, the most relevant takeaway is that “gamification” is not a single mechanism or an “all-purpose” solution. When students rate badges or points as unimportant, this does not imply that the underlying design failed; rather, it indicates that the structural mechanics (quests, progression, task chunking, visibility of next steps) are doing the motivational work, while these surface-

level rewards contribute marginally [8, 35]. This interpretation aligns with systems in which the primary drivers of engagement are clarity, pacing, learning, and feedback rather than competition or extrinsic rewards [34]. Accordingly, OSS-Doorway’s quest structure can be understood as “gamified scaffolding,” in which the central function is to operationalize progression and reduce ambiguity. The related literature suggests that this framing is appropriate as long as the mechanics support mastery-oriented behaviors, such as iteration, the use of feedback, and incremental completion of authentic tasks [35, 26].

2.3 Automated Assessment and Formative Feedback in Programming Education

Automated assessment has become core infrastructure in university programming education because it scales grading and enables rapid, iterative practice. Surveys of automatic assessment systems document a wide range of approaches, including unit-test-based grading, dynamic analysis, and specialized checkers, and highlight that the main educational advantage is providing feedback quickly and consistently across large cohorts [1, 15]. In this setting, feedback is not only summative, but formative, supporting iteration and debugging while students work toward a correct solution.

A consistent theme in the literature is that effectiveness depends on timing, specificity, and alignment with learning goals. Broad syntheses describe formative feedback as most useful when it clarifies target performance, indicates current progress, and provides actionable next steps [12]. Complementary reviews emphasize that feedback should reduce uncertainty about what to do next without replacing the learner’s reasoning, since overly directive feedback can reduce productive struggle or encourage shallow trial-and-error behaviors [28].

Recent empirical work in programming courses has shown that these design choices affect observable student behavior. Leinonen et al. compare immediate feedback to delayed regimes and show that feedback timing can shape students’ submission patterns and engagement during problem-solving [19]. At the same time, systematic reviews show that high-quality automated feedback is challenging, especially when moving beyond binary correctness toward hints, explanations, or strategy-level guidance [16]. As a result, many systems rely on structured, machine-checkable criteria, such as test suites, static checks, and rubric-like validators, because they are more dependable and transparent to students.

Overall, the literature suggests a pragmatic design principle: combine deterministic, criteria-based validation (tests, linters, structured checks) with timely and actionable feedback, and shape the submission loop so that iteration remains learning-oriented rather than purely exploitative [1, 15, 28, 18]. This framing is directly relevant to GitHub-centered workflows, where automated checks (CI, linters, and validators) can serve as formative feedback channels, and where policy knobs (attempt limits, required evidence, or staged milestones) can encourage productive revision behavior.

2.4 LLMs for Learning Support and Feedback

LLMs introduce an interactive, personalized layer of support for students by enabling natural-language explanations, hinting, feedback generation, error diagnosis, clarification, and step-by-step guidance at scale. Recent studies evaluate GPT-like models for generating feedback and hints on programming tasks and measure how this support affects student outcomes across attempts [20, 29]. This shows that LLMs can provide immediate, on-demand guidance (e.g., explanations or debugging-oriented hints), but that the quality of this guidance is variable [29].

Empirical studies also surface clear limitations. In Pankiewicz and Baker’s controlled experiment, students benefited from GPT-provided feedback, but when it was later disabled, they struggled with more complex tasks, suggesting a short-term over-reliance on the model’s guidance [20]. Complementing this, Silva and Costa’s evaluation of four LLMs on a benchmark of student programming solutions shows that a substantial portion of generated feedback is imperfect: while 63% of hints were accurate and complete, 37% contained issues such as incorrect line identification, flawed explanations, or hallucinated problems [29]. All of these indicate that LLM feedback can be helpful, but its reliability varies and its availability can shape how students engage with other feedback channels, which is important to consider when designing automated validation and feedback workflows [20, 29]. For OSS contexts specifically, a key challenge is evaluating conversational agents on realistic, project-specific newcomer questions rather than synthetic prompts. To enable more realistic evaluation of conversational agents for newcomer support in OSS, recent work has also curated datasets of real newcomer questions drawn from project discussions, motivated by the maintainer burden of repeatedly answering similar questions [10].

A qualitative evaluation of Copilot suggests it can change programming workflows by enabling fast, iterative suggestion cycling, requiring users to phrase revision requests precisely, and pushing them to selectively accept or combine suggestions [37]. In educational settings, this can be beneficial if it encourages exploration and refinement, but harmful if it replaces reasoning or obscures assessment criteria. Therefore, designing LLM-based feedback requires careful alignment with learning objectives and transparent expectations.

3 System Design

OSS-Doorway is a GitHub-based coursework system that uses a bot to deliver quests (multi-step learning modules) through GitHub repositories and issues, with automated validation and feedback. The system keeps students within GitHub to reduce context switching, provides structured onboarding to OSS workflows, and gives instructors a configurable pipeline for authoring and deploying learning content.

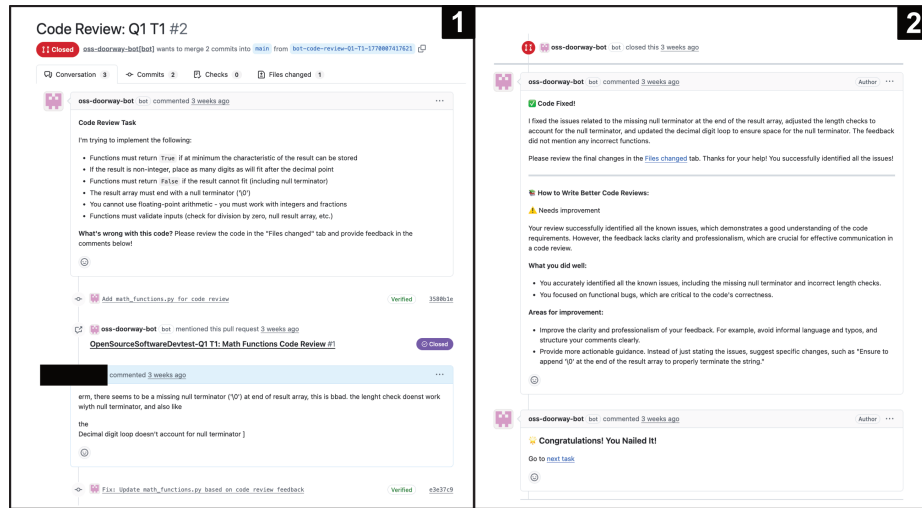


Fig. 1. OSS-Doorway interaction during a code review task. Panel (1) shows a GitHub pull request in which the bot presents the task requirements and asks the student to identify functional issues in the code. Panel (2) shows the bot's follow-up feedback after the student's response, including confirmation of the fixes, formative feedback on the review quality, completion of the task, and guidance toward the next task.

3.1 System Overview and Workflow

OSS-Doorway embeds course activities within a quest-to-task progression implemented via GitHub issues. Each task includes a prompt, an acceptance condition, and a validation strategy (repository metadata checks, multiple-choice questions, pull-request validation, code-review evidence, or artifact/link submission checks). After validation, the system provides immediate feedback and unlocks subsequent tasks based on prerequisites. OSS-Doorway structures course-work using quests (themed learning units) containing tasks (atomic work units). Progression is enforced at both levels, with successful validation unlocking subsequent content.

The system supports three roles: students, instructors, and the system (bot and validator services). A typical student workflow follows: (1) task discovery through repository structure, (2) submission via issue comments, (3) automated validation using deterministic checks or AI-powered text validation, (4) feedback posted to issue threads enabling iterative resubmission, and (5) progression unlocking subsequent tasks upon success. Instructors author content through the Quest Builder, with draft and production configurations separated to enable safe testing during active courses.

3.2 Core Components

The Quest Builder is a React-based web application for quest authoring and configuration management. The backend is a Node.js/Express REST API pro-

viding configuration storage, deployment orchestration, analytics, and repository management.

GitHub serves as the primary interaction surface. Tasks are represented as GitHub issue threads, with validation triggered by comment events, repository events, scheduled checks, or pull-request activity. The system integrates through repository provisioning, issue-based task delivery, comment-based submission, webhook-triggered validation, and automated progression updates.

The validator service supports both deterministic validation (exact matching for multiple-choice tasks, API-backed checks for repository metadata) and text validation using an LLM prompted for educational assessment. During validation, the assistant serves as a “compassionate educational assessment AI” that accepts answers that demonstrate understanding, even with minor formatting issues or alternative wording, and provides supportive feedback rather than simply rejecting incorrect answers.

4 Study Design

We evaluated OSS-Doorway using a post-deployment survey designed to capture students’ perceptions of (i) readiness to contribute to open source software communities, (ii) self-efficacy for GitHub workflows, (iii) task scaffolding, tool acceptance, and overall satisfaction with the system, (iv) quality and usefulness of the built-in validator’s feedback, (v) engagement and enjoyment while completing quests, and (vi) perceived value of the quest experience and intention to contribute to OSS beyond the course.

4.1 Context and Procedure

OSS-Doorway was deployed as part of a mixed undergraduate/graduate course with 30 enrolled students (22 undergraduate and 8 graduate). The instructor designed seven quests (Q1–Q7) spanning Git and GitHub fundamentals, including pull requests/code reviews, and GitHub Actions. A total of $N = 21$ students completed the post-course survey (a 70% completion rate, 21/30), which forms the primary data source for the results reported in this paper.

4.2 Measures

The survey measured multiple constructs. Motivation and internalization were assessed using questions grounded in self-determination theory [6] and cognitive evaluation theory [24], covering interest/enjoyment, perceived competence, and value/usefulness. Because OSS-Doorway was a required assignment, we excluded autonomy/choice items from the theory and focused on internalization. We assessed GitHub self-efficacy (confidence in performing core actions) and OSS readiness using Legitimate Peripheral Participation (LPP)-oriented items (e.g., understanding OSS workflows, finding suitable issues, social norms, and ability to explain a first contribution) [17]. We also measured tool acceptance via

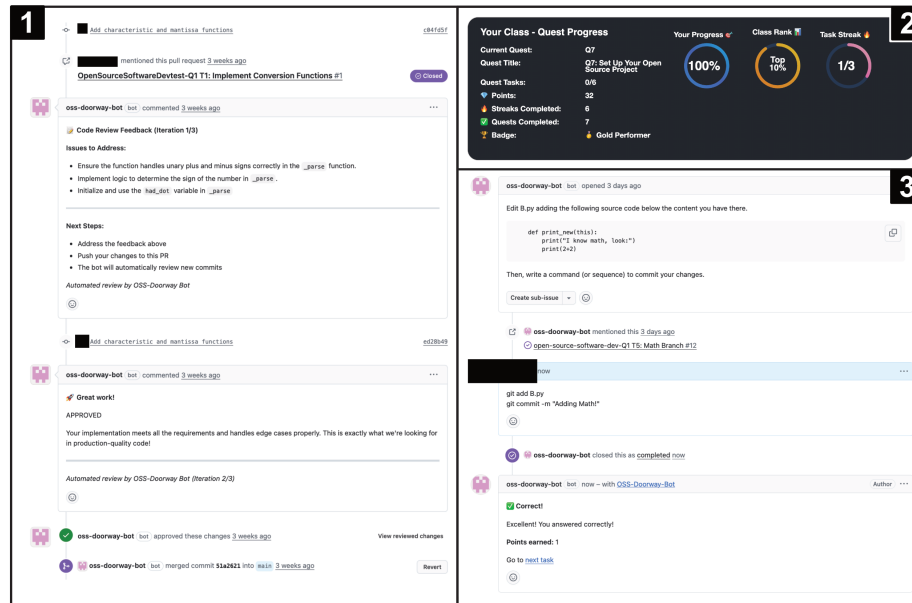


Fig. 2. Examples of OSS-Doorway activities and progress tracking. Panel (1) shows an iterative pull-request review in which the bot identifies issues, provides revision guidance, and approves the pull request after the requested changes are made. Panel (2) shows the Quest Progress dashboard, summarizing the student's current quest, overall progress, rank, streaks, points, completed quests, and badges. Panel (3) shows an automated task-verification workflow in GitHub, including task instructions, the student's command-line actions, completion confirmation, points awarded, and directions for the next task.

TAM/UTAUT-style questions [5, 36], and overall recommendation with an NPS-style item [21]. Engagement and enjoyment were assessed using Likert questions on progress motivation, gamification engagement, and enjoyment of completing quests. Perceived value and intent to contribute were assessed using items on the usefulness of the quests for learning OSS contribution, increased likelihood of contributing, and intention to contribute outside the course. Task scaffolding items assessed the clarity and actionability of guidance (including navigation via the README structure), and validator-focused items assessed whether feedback and hints supported revision, identified mistakes, and provided actionable next steps, along with a global quality rating and open-ended feedback. Finally, we collected open-ended reflections (most/least helpful, frustrations, and suggestions) and demographics, including course level, professional experience, prior GitHub/OSS experience, and gender.

4.3 Analysis Approach

We analyze Likert-style responses using descriptive statistics and report construct-level composite scores. For multi-item constructs, we compute subscale means by averaging items within each subscale and report internal consistency using Cronbach’s α [4].

Open-ended responses are analyzed using an inductive thematic analysis [3]. We report themes with representative excerpts and triangulate qualitative findings with the quantitative constructs (e.g., linking validator comments to validator Likert ratings).

5 Results

This section reports results from the post-course survey for CS499/599 OSS-Doorway ($N = 21$) and summarizes open-ended feedback. We summarize results for (i) perceived OSS readiness, (ii) GitHub self-efficacy, (iii) task scaffolding and tool experience, (iv) validator feedback quality, (v) engagement and enjoyment, and (vi) perceived value and intent to contribute, after which we provide our overall analysis.

5.1 Perceived OSS Readiness

Respondents generally reported moderately positive perceived readiness for OSS participation and confidence with core GitHub workflows. Across the OSS readiness set of questions (understanding the typical OSS workflow, feeling prepared to make a meaningful contribution, feeling less intimidated about contributing), composite ratings were positive overall (mean = 5.86 out of 7, SD = 0.84; Table 1, Figure 4). The highest-rated item was understanding the typical OSS workflow (mean = 6.10, SD = 0.89), suggesting that the quest structure helped students internalize the sequence of steps involved in contributing. Ratings were also positive for feeling less intimidated about contributing (mean = 5.76, SD =

Table 1. Survey item and composite statistics for students' perceptions of OSS-Doorway.

Composite & items	Mean	SD	α
(i) Perceived OSS readiness (1–7)			
I understand the typical OSS workflow	6.10	0.89	
I feel prepared to make a meaningful contribution to OSS	5.71	0.85	
I feel less intimidated about contributing to OSS	5.76	1.04	
<i>Composite</i>	5.86	0.84	.888
(ii) GitHub self-efficacy (1–5)			
Confident: use GitHub issue tracker to find open issues	4.81	0.51	
Confident: understand GitHub pull-requests	4.48	0.81	
Confident: open a pull request using GitHub web interface	4.71	0.64	
Confident: use GitHub to contribute to projects	4.38	0.80	
<i>Composite</i>	4.60	0.59	.856
(iii) Task scaffolding and tool experience (1–7)			
Tasks improved my understanding of class requirements	5.62	1.50	
Tasks gave more actionable guidance than standard assignments	5.76	1.58	
Using OSS-Doorway was easy for me	6.00	1.38	
OSS-Doorway is useful to learn how to contribute to OSS	5.48	1.50	
<i>Composite</i>	5.71	1.35	.926
(iv) Validator feedback quality (1–5)			
Validator's feedback influenced how I adjusted my answers	4.16	0.90	
Validator helped me identify specific mistakes	4.21	0.79	
<i>Composite</i>	4.18	0.79	.844
(v) Engagement and enjoyment (1–7)			
Seeing my progress through quests motivated me to continue	5.43	1.36	
Gamification made the learning experience more engaging	4.57	1.75	
I enjoyed completing quests in OSS-Doorway very much	4.71	1.55	
<i>Composite</i>	4.90	1.31	.791
(vi) Perceived value and intent to contribute (1–7)			
Doing the quests is useful for learning OSS contribution	5.38	1.43	
I am more likely to contribute to OSS now than before	6.33	0.66	
After this experience, I intend to contribute to OSS outside class	5.86	1.06	
<i>Composite</i>	5.86	0.68	.198
Recommendation (single item)			
Likelihood to recommend OSS-Doorway (0–10)	7.33	2.22	—
NPS (%Promoters - %Detractors)	+4.8	—	—

Note. Item rows report mean and standard deviation. Composite rows report the average across items in each construct, standard deviation, and Cronbach's α when applicable. Response scales vary by construct and are shown in parentheses. The perceived value and intent-to-contribute items are reported together for descriptive purposes but should be interpreted item by item because of low internal consistency.

1.04). Overall, students reported feeling comfortable with the main mechanics of contribution workflows after the quest sequence, although the size of the sample and reliance on self-report limit how strongly we can interpret these ratings.

5.2 GitHub Self-Efficacy

Students reported relatively high confidence across the four GitHub-specific tasks (composite mean = 4.60 out of 5, SD = 0.59; Table 1, Figure 3). Confidence in using the issue tracker to find open issues scored highest at 4.81, followed by opening a pull request via the GitHub web interface (4.71). Understanding pull requests (4.48) and using GitHub to contribute to projects more broadly (4.38) were the lowest-rated items, but both were above 3 (the midpoint). These results suggest that after the quest sequence, students felt reasonably confident performing the specific mechanics of a GitHub-based contribution workflow, though the lower ratings on broader contribution tasks indicate that confidence is uneven across activities.

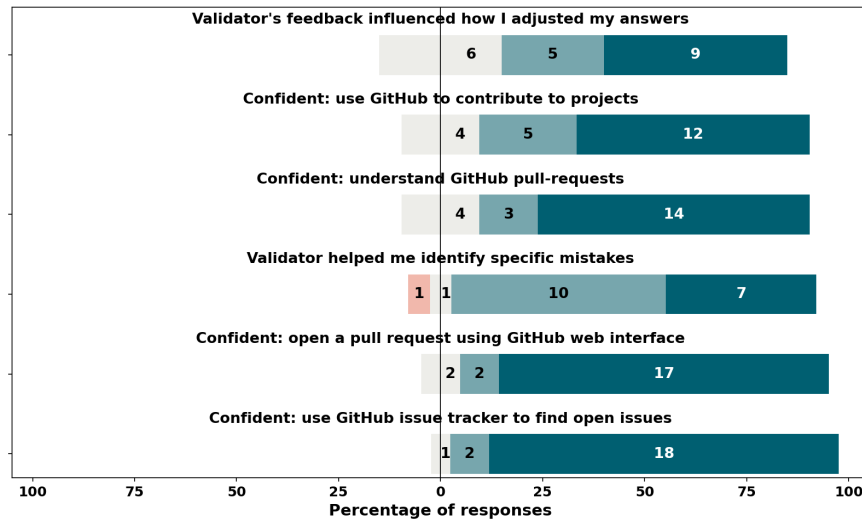


Fig. 3. Student ratings of GitHub self-efficacy and validator feedback quality items on a 5-point scale ($N = 19-21$).

5.3 Task Scaffolding and Tool Experience

Students rated task-level instructional scaffolding and the overall tool moderately positively (composite mean = 5.71 out of 7, SD = 1.35; Table 1, Figure 4). Items about tasks improving understanding of class requirements and providing more actionable guidance than standard assignments scored among the highest, suggesting that the quest-to-task structure helped make expectations concrete.

Perceptions of overall tool experience were generally favorable, with students reporting that OSS-Doorway was reasonably easy to use. Students also indicated that OSS-Doorway was useful for learning how to contribute to OSS (mean = 5.48 out of 7, SD = 1.50). The relatively large standard deviations on these items, however, indicate that experiences varied noticeably across respondents.

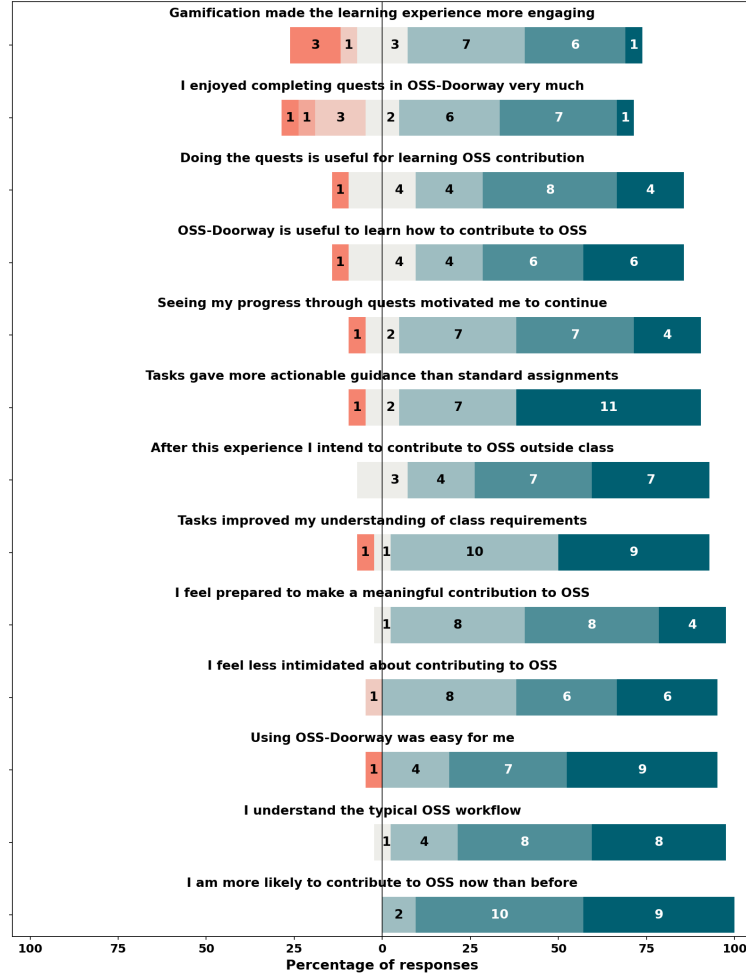


Fig. 4. Student agreement ratings for OSS readiness, task scaffolding and tool experience, engagement and enjoyment, and perceived value and intent to contribute on a 7-point scale ($N = 21$).

5.4 Validator Feedback Quality

Validator feedback is included in the broader construct of quest structure, tasks, and the built-in validator (composite mean = 4.18 out of 5, SD = 0.79; Table 1,

Figure 3). The two validator items (whether feedback influenced how students adjusted their answers and whether the validator helped identify specific mistakes) contribute to the overall score. This aligns with a recurring theme found in the open-ended feedback, where students described cases in which rejection messages were difficult to act on.

5.5 Engagement and Enjoyment

Students rated engagement and enjoyment with the system positively but with mixed enthusiasm (composite mean = 4.90 out of 7, $SD = 1.31$, $\alpha = .791$; Table 1, Figure 4). The construct combines items on progress motivation, perceived engagement of the gamification layer, and enjoyment of completing quests. The composite sits closer to the midpoint than to “agree,” and the “gamification made the learning experience more engaging” item itself averaged only 4.57 ($SD = 1.75$), which is closer to neutral and exhibits substantial variability, indicating that the surface gamification layer was received with mixed enthusiasm.

5.6 Perceived Value and Intent to Contribute

Perceived value and intent to contribute combines items on the usefulness of the quests for learning OSS, increased likelihood of contributing, and explicit intent to contribute outside the course. Item-level means are uniformly positive (5.38, 6.33, and 5.86 out of 7; Table 1, Figure 4) and the composite mean is 5.86 ($SD = 0.68$); however, internal consistency is poor ($\alpha = .198$), so the three items should be interpreted individually rather than as a single coherent scale. Students endorsed each statement on its own, but the items appear to capture related yet distinct ideas (utility of the activity for learning, perceived behavior change, and forward-looking intent) rather than a single underlying construct.

5.7 Overall Recommendation

We measured overall recommendation using a Net Promoter Score (NPS) item (0–10). Following standard NPS conventions, scores of 9–10 are Promoters, 7–8 are Passives, and 0–6 are Detractors. In our sample ($N=21$), 23.8% were Promoters (5/21), 57.1% were Passives (12/21), and 19.0% were Detractors (4/21), yielding an NPS of +4.8 ($\approx +5$; see Figure 5). The raw recommendation ratings had $M = 7.33$ and $SD = 2.22$. We interpret these numbers cautiously. While the NPS is positive, it is only marginally so: most respondents fell into the Passive band, and nearly one in five was a Detractor, with the score sitting well below levels associated with strong advocacy. We therefore read the recommendation signal as lukewarm rather than enthusiastic, consistent with the friction points reported in the survey, particularly around validator reliability.

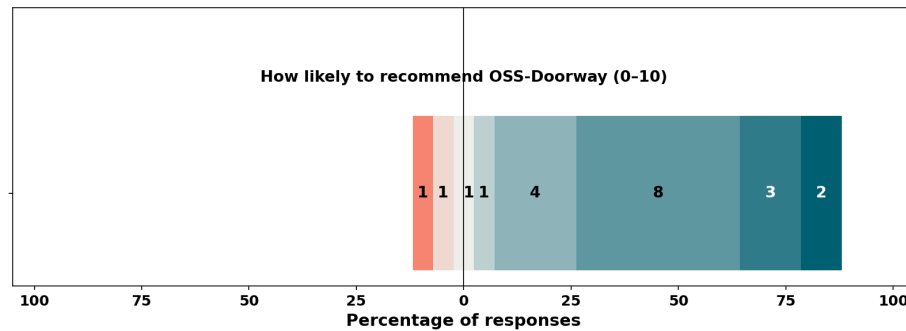


Fig. 5. Students’ likelihood to recommend OSS-Doorway to a peer, measured with a 0–10 Net Promoter Score item ($N = 21$). Scores of 9–10 are classified as Promoters, 7–8 as Passives, and 0–6 as Detractors.

5.8 Qualitative Feedback from Open-Ended Survey Responses

Open-ended responses were analyzed inductively and yielded six themes. The qualitative feedback aligned with the survey results: students valued the GitHub-based experience, hands-on workflow and immediate feedback, but reported friction around the validator reliability, feedback specificity, and onboarding clarity.

Hands-on, GitHub-based learning was perceived as valuable. A consistent positive theme was that working directly within GitHub, rather than through a traditional LMS, made the material feel more authentic and actionable for many students. Several respondents contrasted the experience favorably against conventional assignments and noted that the platform encouraged active participation rather than passive reading.

Student feedback

The general path that the Doorway bot required me to follow benefited my learning the most, as I was forced to physically work through the process. This helps me to remember and understand how to do things better than learning about it from a high level.

I think this was a lot more fun. I get kind of sick {with} canvas assignments and needing to tab back and forth, so having everything in one place was great.

...I think OSS-Doorway is ESSENTIAL in order to help new students understand GitHub. As such, it helped it a lot, and I think we should move it into an introductory class to teach students GitHub early, allowing them to save their code and get started in OSS maybe in Freshman year instead of the Senior one.

Progressive quest structure reduced ambiguity and sustained forward momentum. Students frequently described the quest-based progression as clarifying “what to do next” and enabling focused, incremental learning. The step-by-step structure was viewed as particularly valuable for avoiding the cognitive overload that can arise when students are left to self-navigate complex GitHub workflows.

Student feedback

The progressive structure guides step by step, focusing strictly on an area before moving on to more content.

It felt like it was a very guided exercise, which is good for beginners getting their first introduction.

The activity as a whole was well structured in a way that actually helped me learn.

I think the guiding of the tasks was very useful and much more fun than navigating through canvas to do the same tasks.

Immediate feedback was valued, in principle. Students appreciated receiving instant confirmation and guidance without waiting for manual grading, describing it as making the quests easier to complete on their own schedule and supporting faster iteration through multiple attempts. As discussed below, however, the value of this loop was tempered when the validator itself behaved unpredictably.

Student feedback Having immediate feedback is very nice. It really allowed me to work on quests at any time and receive live feedback immediately.

Without the bot, I would have spent more time on each assignment making sure my submission was perfect before I would submit. But in this environment, I could just try the first thing that came to mind because I had multiple attempts.

Validator reliability and format sensitivity were the primary pain points. The most common frustration across both the validator-specific and free-response questions was that the validator sometimes rejected correct answers unless they conformed to a very specific format, and that its rejection messages were not always actionable or diagnostic. Several students noted that the system could be gamed in the opposite direction as well, with the validator occasionally accepting responses that did not reflect genuine understanding.

Student feedback The validator is just not that good, does not give the best feedback, and seems to struggle in validating things that are actually correct but not in the format it wants.

Sometimes, it was the bot not quite understanding my answer that would mess up my badge and/or answer streak. Me saying “The answer is {number}” and getting marked as wrong and losing my streak because all it wanted was “{number}” did mildly irk me.

The validator was awesome... Also it was kind of easy to manipulate the validator a couple of times. I would tell it what I thought was supposed to happen without doing it or doing it wrong and it would still verify the information as correct.

If the validator could give custom advice instead of prepackaged responses, and have a conversation with the user, I think the whole system would be much better. The validator is probably the biggest weakness for OSS-Doorway.

Onboarding clarity and submission conventions need to be made explicit earlier. A recurring source of early confusion was not knowing how to interact with the system in the first place, specifically, that answers should be submitted as issue comments. Students who encountered this confusion reported spending time at the start of Quest 1 before discovering the interaction model independently. Related frustrations arose in the Code Review quest, where submission expectations and feedback location were unclear to multiple respondents.

Student feedback When I first started, I wasn’t really sure what to do and so I sat there for a while trying to figure out how to start the quest. Eventually, I realized I just sent my submission as a comment.

On quest 1, task 1, maybe just add one line saying “Send your answer as a comment here.” And that’s about it. It would have stopped my confusion at the very start about where exactly I was supposed to answer.

The most confusion for me came from the Code Review quest, as it was unclear where the feedback would be given, if feedback was even given, if what was being submitted was correct, etc.

Gamification structure was useful; surface rewards were largely ignored. Responses to the gamification prompt revealed a consistent split: students appreciated the quest and progression structure as an organizational mechanism, while points, badges, and streaks were seen as largely irrelevant or even counterproductive when tied to validator inconsistency. Several students reframed the gamification dashboard not as a motivational tool but as a progress tracker, suggesting its primary utility is informational rather than competitive.

Student feedback I never really felt like it was a game. I used the statistics more as a dashboard to gauge my own progress. I think the gamification elements could be built on to make a more interactive/useful dashboard, but the gamification was not of huge value to me.

I like the split into quests, but that’s about it -- medals, badges, and points meant nothing to me, ...

They are good, but the motivation to do homework, at the end of the day, always comes back to grades. Also, because the whole class is more or less in the same place, being individualized by the gamified metrics is not really possible.

The open-ended responses sharpen and extend the quantitative picture. OSS-Doorway’s strongest perceived contributions are its GitHub-based structure, hands-on task design, and real-time feedback availability.

6 Discussion

Our findings reveal both the promise and the current limitations of OSS-Doorway as a GitHub-based, gamified coursework system with automated feedback. We discuss the implications for scalable onboarding and feedback in software engineering education.

6.1 GitHub-Based Structure as an Onboarding Mechanism

The clearest pattern across both quantitative and qualitative data is that OSS-Doorway’s GitHub-based structure reduced friction and made authentic contribution workflows feel accessible. High composite scores on OSS readiness and GitHub self-efficacy (combined $M = 4.60/5$), along with comments of the system being “ESSENTIAL” for learning GitHub and a recommendation to deploy it earlier in the curriculum, suggest that the system’s value extends beyond skill acquisition to confidence-building and workflow socialization. This aligns with prior work showing that newcomers benefit not only from technical scaffolding but also from guided, repeated participation in the same interaction patterns they will encounter in real repositories [31, 27].

6.2 Structured Progression as the Primary Engagement Driver

Students consistently framed engagement in terms of structure rather than reward. The quest-to-task progression was described as keeping work focused, reducing uncertainty about next steps, and enabling incremental skill-building in a way that felt more engaging than conventional assignments. Importantly, students separated the quest structure from the reward layer: students who reported finding badges and points irrelevant still described the quest structure positively, and several reframed the progress dashboard as an informational tracker rather than a motivational tool.

This pattern is consistent with literature on gamification, which distinguishes between structural mechanics (progression, task chunking, and visible milestones) and surface rewards (points, badges, and leaderboards), and finds that the former more reliably supports learning-oriented behavior [8, 34, 35]. For OSS-Doorway, this means the quest structure matters the most, while points and badges contribute only marginally.

6.3 The Validator as the Primary Improvement Opportunity

The validator is the clearest area for improvement and the one most directly tied to student experience. It received a lower composite score ($M = 4.18/5$) and the most consistent negative qualitative feedback, with one of the weakest individual item ratings, being how much the validator’s feedback influenced adjustment of students’ answers ($M = 4.16$).

The qualitative data adds important information to these numbers. Students identified two distinct failure modes. The first is false rejection: correct answers being marked wrong due to formatting rather than conceptual error (e.g., writing “The answer is {number}” instead of “{number}”). The second is false acceptance: the validator approving responses that described the correct behavior without the student actually performing it. Both failure modes undermine the validator’s core function; the first by frustrating students who are right, and the second by allowing students to bypass genuine learning. This suggests that additional work may be done in the LLM prompting to improve the right level of assessment.

6.4 Automation as a Feedback Assistant, Not an Opaque Judge

Automated feedback is most effective when its logic is legible to learners. When students understand what is being checked and why an answer was rejected, they can act on the feedback. When they cannot, they experience the system as measuring compliance with hidden preferences rather than conceptual understanding. This was a frustrating and unfortunate dynamic that several students described explicitly. This is consistent with broader findings in formative feedback research, which emphasize that feedback must reduce uncertainty about what to do next without replacing the learner’s reasoning [28, 12].

For OSS-Doorway, this principle argues for positioning the automated validator as a *feedback assistant* rather than a final authority. In practice, this means

making explicit the criteria being checked (“your answer should include X, Y, and Z”), distinguishing the type of failure (missing concept vs. formatting vs. wrong artifact), and anchoring acceptance decisions in explicit, rule-based logic wherever possible. It also means openly acknowledging the limits of LLM-based judgment: when the system is uncertain, it should state so and prompt human review rather than issuing a confident but unreliable verdict.

6.5 Implications for GitHub-Centered Software Engineering Coursework

Our findings suggest some actionable implications for GitHub-centered educational systems. First, GitHub-based interaction can reduce context switching and support authentic workflow socialization, but submission norms must be made explicit at every entry point. Second, structured progression and task chunking are the primary engagement mechanisms; at the same time, surface reward layers should remain secondary. Third, validator improvements should prioritize criterion transparency; deterministic checks paired with clear rubrics outperform a capable but unpredictable model. Fourth, the system shows genuine potential for deployment in the curriculum, with students independently identifying that OSS-Doorway’s onboarding value would be higher in introductory courses, where GitHub familiarity cannot be assumed.

Overall, OSS-Doorway appears to be a viable model for OSS education, with quantitative and qualitative results pointing to the value of structured guidance within GitHub. However, the moderate recommendation score shows that the current implementation is not yet uniformly well received, with validator reliability and feedback interpretability remaining the clearest areas for improvement.

7 Limitations

This study has several limitations that should be considered when interpreting the results and when building on current work.

Although the evaluation was conducted in an open-source-focused course with a mixed cohort of graduate/undergraduate students, which provided a participant pool with varied experience in personal and industry projects, it was still limited to a single course with a relatively small sample size. While this setting reflects realistic classroom use, the findings may not generalize to other courses, institutions, or student populations.

Secondly, the results rely primarily on self-reported survey data, which may be subject to perception bias. Students’ reported experiences may not perfectly reflect objective learning outcomes or long-term skill acquisition. Although system usage artifacts were collected, this study does not include controlled pre-post measures or external performance benchmarks that would allow us to quantify improvement over time and compare it against a consistent reference. As a result, the findings should be interpreted primarily as evidence about student perceptions, rather than definitive measures of learning impact.

In addition, the study does not include a controlled comparison with alternative instructional approaches such as traditional assignments or even other GitHub-based tooling. As a result, any causal claims about the system’s superiority should be avoided. Our system was designed to be fully customizable, enabling instructors to tailor a course module. In light of this, the effectiveness of the automated validation and feedback mechanism is partly dependent on the quality of task authoring, including prompt clarity, validation criteria, and feedback templates. Variability in task design may influence students’ experiences not only with feedback quality and clarity but also with the system as a whole. Finally, access to external large language models (LLMs) was not controlled or restricted during the course. Students’ use of such tools may have influenced how they interpreted feedback or completed tasks, introducing an additional factor that was not fully isolated in the current study.

A further concern is the potential for social desirability bias. Even though responses were collected anonymously and reported only in aggregate, students may have skewed their answers toward more favorable assessments to avoid implicitly criticizing the tool, which could inflate ratings on subjective items such as engagement, perceived usefulness, and overall recommendation.

Beyond the role of task-authoring quality noted above, OSS-Doorway’s reliance on the Quest Builder makes the student experience strongly instructor-dependent in ways that limit the generality of the results. The same software can yield meaningfully different outcomes when instructors vary task wording, validator prompts, acceptance criteria, retry policies, or pacing across quests. Our findings should therefore be read as an evaluation of one particular configuration of OSS-Doorway rather than of the system in the abstract. Future evaluations could mitigate this by reporting the full quest configuration as part of the replication package and by comparing outcomes across deployments that share a common configuration baseline.

Finally, the validator’s reliance on a hosted large language model introduces a model-drift and replicability concern that the current paper does not fully address. We do not report the specific model identifier, version, or prompt parameters (e.g., temperature, top- p , system message, and few-shot exemplars) of the validator in enough detail to support exact replication, and the underlying model can also be silently updated by its provider over time. Because validator behavior is sensitive to all of these factors, results obtained with one configuration may not be exactly reproducible in a different time window or with a different model snapshot. Future iterations of the system should pin and document model versions, log all prompt parameters alongside validator decisions, and include the relevant configuration in the replication package.

8 Future Work

Building on a key limitation of our current integration and evaluation, a promising direction is to extend OSS-Doorway beyond a single course offering. Future work should evaluate OSS-Doorway beyond a single course. Deployments across

courses, institutions, academic levels, and student populations would improve external validity and help compare its use across teaching contexts.

Future studies should also include a baseline or control condition. The current evaluation relies on one cohort and self-reported perceptions. A stronger design could compare OSS-Doorway with a prior version of the same course that does not use the bot. Objective measures such as grades, assignment completion, pull requests, and task completion time would help distinguish the effects of OSS-Doorway from course, instructor, or cohort effects.

We also plan two system improvements. First, we will add a short onboarding tutorial before Quest 1 to explain how students should interact with the bot and submit answers. Second, we will improve the validator using examples from real student responses to reduce format sensitivity and make feedback clearer and more useful.

The validator also needs a separate evaluation. Students reported problems with incorrect rejections and incorrect approvals, but this study does not measure validator accuracy. We plan to log validator decisions and compare them with human judgments to estimate accuracy across task types.

Future evaluations should measure learning and contribution behavior, not only perceptions. For example, a follow-up study one or two months after the course could combine a short survey with GitHub activity data to examine whether students continue using GitHub or contributing to OSS after the course.

Finally, students suggested that OSS-Doorway may be useful earlier in the curriculum, where GitHub experience cannot be assumed. Future work should evaluate the system in introductory software engineering or computing courses to assess its value for students with less experience using version control.

9 Conclusion

This paper presents *OSS-Doorway*, a learning system that combines gamified elements to improve retention, a GitHub-based approach that introduces students to the platform, and a feedback system aimed at onboarding students into OSS contributions. By using structured tasks, gamified mechanisms, automated validation, and iterative feedback delivered entirely within GitHub, we can keep students in authentic open-source workflows, thereby supporting onboarding into authentic software engineering and open-source workflows. Traditional coursework is transformed by organizing it as progressive quests and embedding interactions within GitHub artifacts, such as repositories and issues.

OSS-Doorway’s classroom deployment demonstrates that this approach is viable and beneficial in several key respects. Through quantitative and qualitative analyses, we find that students improved their familiarity with GitHub workflows. Also, our quest-based structure and explicit progression reduced ambiguity about what to do next and contributed to higher perceived engagement. These results suggest that gamified systems on GitHub, such as OSS-Doorway, can support learning and onboarding in open source education, but their effectiveness

depends strongly on the quality and transparency of automated feedback mechanisms. By addressing current feedback limitations and extending evaluation across contexts, such systems have the potential to scale authentic, feedback-rich learning experiences within the platforms students will encounter in real-world software development.

Data Availability

The replication package, with the research instruments and scripts, is /available for public access [9]. Likewise the bot and related management system are also publicly available through GitHub [22, 23].

Acknowledgements

We thank the students who took the course and shared their feedback on OSS-Doorway. This work was partially supported by the U.S. National Science Foundation under Award #2247929.

References

1. Ala-Mutka, K.M.: A survey of automated assessment approaches for programming assignments. *Computer Science Education* **15**(2), 83–102 (2005). <https://doi.org/10.1080/08993400500150747>
2. Balali, S., Steinmacher, I., Annamalai, U., Sarma, A., Gerosa, M.A.: Newcomers' barriers. . . is that all? an analysis of mentors' and newcomers' barriers in oss projects. *Computer Supported Cooperative Work (CSCW)* **27** (12 2018). <https://doi.org/10.1007/s10606-018-9310-8>
3. Braun, V., Clarke, V.: Using thematic analysis in psychology. *Qualitative Research in Psychology* **3**(2), 77–101 (2006). <https://doi.org/10.1191/1478088706qp063oa>
4. Cronbach, L.J.: Coefficient alpha and the internal structure of tests. *Psychometrika* **16**(3), 297–334 (1951). <https://doi.org/10.1007/BF02310555>
5. Davis, F.D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly* **13**(3), 319–340 (1989), <http://www.jstor.org/stable/249008>
6. Deci, E.L., Ryan, R.M.: The "what" and "why" of goal pursuits: Human needs and the self-determination of behavior. *Psychological Inquiry* **11**(4), 227–268 (2000). https://doi.org/10.1207/S15327965PLI1104_01
7. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From game design elements to gamefulness: defining "gamification". In: 15th International Academic MindTrek Conference: Envisioning Future Media Environments. p. 9–15. MindTrek'11, ACM, New York, NY, USA (2011). <https://doi.org/10.1145/2181037.2181040>
8. Dicheva, D., Dichev, C., Agre, G., Angelova, G.: Gamification in education: A systematic mapping study. *Journal of Educational Technology & Society* **18**(3), 75–88 (2015), <http://www.jstor.org/stable/jeductechsoci.18.3.75>
9. Etchie, M., Smallwood, K., Gerosa, M., Steinmacher, I.: OSS-Doorway replication package (research instruments and scripts). Zenodo (2026), <https://zenodo.org/records/20709954>, replication package

10. Etchie, M.P., Beach, H., Felizardo, K., Steinmacher, I.: Towards a dataset to assess conversational agent's efficacy in mentoring newcomer developers. In: 2025 IEEE/ACM International Workshop on Bots in Software Engineering (BotSE). p. 48–52. IEEE Press (2025). <https://doi.org/10.1109/BotSE67031.2025.00018>
11. Feliciano, J., Storey, M.A., Zagalsky, A.: Student experiences using github in software engineering courses: a case study. In: 38th International Conference on Software Engineering Companion. p. 422–431. ICSE '16, ACM, New York, NY, USA (2016). <https://doi.org/10.1145/2889160.2889195>
12. Hattie, J., Timperley, H.: The power of feedback. *Review of Educational Research* **77**(1), 81–112 (2007). <https://doi.org/10.3102/003465430298487>
13. He, H., Su, H., Xiao, W., He, R., Zhou, M.: GFI-bot: automated good first issue recommendation on GitHub. In: 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. p. 1751–1755. ESEC/FSE 2022, ACM, New York, NY, USA (2022). <https://doi.org/10.1145/3540250.3558922>
14. Hsing, C., Gennarelli, V.: Using GitHub in the classroom predicts student learning outcomes and classroom experiences: Findings from a survey of students and teachers. In: 50th ACM Technical Symposium on Computer Science Education. p. 672–678. SIGCSE '19, ACM, New York, NY, USA (2019). <https://doi.org/10.1145/3287324.3287460>
15. Ihantola, P., Ahoniemi, T., Karavirta, V., Seppälä, O.: Review of recent systems for automatic assessment of programming assignments. In: 10th Koli Calling International Conference on Computing Education Research. p. 86–93. Koli Calling '10, ACM, New York, NY, USA (2010). <https://doi.org/10.1145/1930464.1930480>
16. Keuning, H., Jeuring, J., Heeren, B.: A systematic literature review of automated feedback generation for programming exercises. *ACM Trans. Comput. Educ.* **19**(1) (Sep 2018). <https://doi.org/10.1145/3231711>
17. Lave, J., Wenger, E.: *Situated Learning: Legitimate Peripheral Participation*. Cambridge university press (1991)
18. Lawrence, R., Foss, S., Urazova, T.: Evaluation of submission limits and regression penalties to improve student behavior with automatic assessment systems. *ACM Trans. Comput. Educ.* **23**(3) (Jun 2023). <https://doi.org/10.1145/3591210>
19. Leinonen, J., Denny, P., Whalley, J.: A comparison of immediate and scheduled feedback in introductory programming projects. In: 53rd ACM Technical Symposium on Computer Science Education - Volume 1. p. 885–891. SIGCSE 2022, ACM, New York, NY, USA (2022). <https://doi.org/10.1145/3478431.3499372>
20. Pankiewicz, M., Baker, R.S.: Large language models (gpt) for automating feedback on programming assignments. In: 31st International Conference on Computers in Education. Asia-Pacific Society for Computers in Education (2023), <https://arxiv.org/abs/2307.00150>
21. Reichheld, F.F.: The one number you need to grow. *Harvard Business Review* **81**(12), 46–55 (2003), <https://hbr.org/2003/12/the-one-number-you-need-to-grow>
22. RESHAPE Lab: OSS-Doorway (2026), <https://github.com/RESHAPELab/OSS-Doorway>
23. RESHAPE Lab: OSS-Management (2026), <https://github.com/RESHAPELab/OSS-Management/tree/development>
24. Ryan, R.: Control and information in the intrapersonal sphere: An extension of cognitive evaluation theory. *Journal of Personality and Social Psychology* **43**, 450–461 (09 1982). <https://doi.org/10.1037/0022-3514.43.3.450>

25. Sarma, A., Gerosa, M.A., Steinmacher, I., Leano, R.: Training the future workforce through task curation in an oss ecosystem (2016). <https://doi.org/10.1145/2950290.2983984>
26. Sheth, S., Bell, J., Kaiser, G.: Increasing student engagement in software engineering with gamification. Tech. rep., Department of Computer Science, Columbia University (2012), <https://doi.org/10.7916/D8H99DF0>
27. Sholler, D., Steinmacher, I., Ford, D., Averick, M., Hoye, M., Wilson, G.: Ten simple rules for helping newcomers become contributors to open projects. *PLoS Computational Biology* **15**(9) (2019). <https://doi.org/10.1371/journal.pcbi.1007296>
28. Shute, V.J.: Focus on formative feedback. *Review of Educational Research* **78**(1), 153–189 (2008). <https://doi.org/10.3102/0034654307313795>
29. Silva, P., Costa, E.: Assessing large language models for automated feedback generation in learning programming problem solving. *Proceedings of Machine Learning Research* **273**, 1–9 (2025), <https://proceedings.mlr.press/v273/silva25a.html>
30. Steinmacher, I., Conte, T., Gerosa, M.A., Redmiles, D.: Social barriers faced by newcomers placing their first contribution in open source software projects. In: 18th ACM Conference on Computer Supported Cooperative Work & Social Computing. p. 1379–1392. CSCW '15, ACM, New York, NY, USA (2015). <https://doi.org/10.1145/2675133.2675215>
31. Steinmacher, I., Conte, T.U., Treude, C., Gerosa, M.A.: Overcoming open source project entry barriers with a portal for newcomers. In: 38th International Conference on Software Engineering. p. 273–284. ICSE '16, ACM, New York, NY, USA (2016). <https://doi.org/10.1145/2884781.2884806>
32. Steinmacher, I., Graciotto Silva, M.A., Gerosa, M.A., Redmiles, D.F.: A systematic literature review on the barriers faced by newcomers to open source software projects. *Inf. Softw. Technol.* **59**(C), 67–85 (Mar 2015). <https://doi.org/10.1016/j.infsof.2014.11.001>
33. Straubinger, P., Fraser, G.: Improving testing behavior by gamifying intelij. In: IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3597503.3623339>
34. Subhash, S., Cudney, E.A.: Gamified learning in higher education: A systematic review of the literature. *Comput. Hum. Behav.* **87**(C), 192–206 (2018). <https://doi.org/10.1016/j.chb.2018.05.028>
35. Toda, A.M., Klock, A.C., Oliveira, W., Palomino, P.T., Rodrigues, L., Shi, L., Bittencourt, I., Gasparini, I., Isotani, S., Cristea, A.I.: Analysing gamification elements in educational environments using an existing gamification taxonomy. *Smart Learning Environments* **6**(1), 1–14 (2019). <https://doi.org/10.1186/s40561-019-0106-1>
36. Venkatesh, V., Morris, M.G., Davis, G.B., Davis, F.D.: User acceptance of information technology: Toward a unified view. *MIS Quarterly* **27**(3), 425–478 (2003). <https://doi.org/10.2307/30036540>
37. Wermelinger, M.: Using GitHub Copilot to solve simple programming problems. In: 54th ACM Technical Symposium on Computer Science Education. pp. 172–178. SIGCSE 2023, ACM, New York, NY, USA (2023). <https://doi.org/10.1145/3545945.3569830>
38. Xiao, W., He, H., Xu, W., Tan, X., Dong, J., Zhou, M.: Recommending good first issues in github oss projects. In: 44th International Conference on Software Engineering. p. 1830–1842. ICSE '22, ACM, New York, NY, USA (2022). <https://doi.org/10.1145/3510003.3510196>