Journal of Soft Computing Paradigm (ISSN: 2582-2640)
www.irojournals.com/jscp/

CrowdTest: Gamification as Cognitive Support to Improve Bug Report Quality in Crowdsourced Software Testing

Prabin Gautam

Department of Electronics and Computer Engineering, Paschimanchal Campus, Tribhuvan University, Pokhara, Nepal.

Email: prabeen122@gmail.com

Abstract

Non-expert testers often struggle to write clear, reproducible, and useful bug reports. This study examines whether providing cognitive support in crowdsourced software testing improves bug report quality. A web-based prototype, CrowdTest, was used to collect 30 bug reports from anonymous participants, with 15 reports submitted through a non-reward gamified interface and 15 through a baseline interface. Three independent evaluators blindly rated each report on clarity, reproducibility, completeness, and usefulness with acceptable inter-rater reliability for clarity, usefulness, and reproducibility, but lower reliability for completeness. Because the score distributions were non-normal, group differences were analyzed using the Mann-Whitney U test. The non-reward-based gamified condition performed significantly better on clarity, usefulness, and overall composite score, while reproducibility showed a positive but non-significant trend. Completeness is reported for observational purposes because of its lower inter-rater reliability, and the overall composite score should therefore be interpreted with caution. Overall, the results provide preliminary support that the cognitive support may improve the quality of bug reports, although the findings are limited by small sample size, scenario variation, and missing telemetry data.

Keywords: Crowdsourced Testing, Non-Reward Gamification, Cognitive Scaffolding, Bug Report Quality, Non-Expert Testers

DOI: https://doi.org/10.36548/jscp.2026.2.001
Received: 13.02.2026, received in revised form: 14.03.2026, accepted: 31.03.2026, published: 10.04.2026

1. Introduction

In Crowdsourced Software Testing (CST), a group of non-expert users are asked to test an application and report what they find. Unlike traditional testing, which relies heavily on experts working in a strict, controlled environment, CST tests applications in real-world environments using various devices. Platforms such as TestBirds, Applause, and Testlio have made this testing method more accessible and manageable. CST benefits not only from the number of bugs found but also from the quality of the bug reports submitted by the testers. In this study, the quality of a bug report is perceived as how clear, complete, and useful a report is for developers. A high-quality bug report should be 1) clear: easy for developers to understand the description of the bug, and 2) complete: contain all necessary information for developers to understand and reproduce the bug. This usually includes details such as the steps to reproduce the bug, expected result, actual result, and any other context in which the bug was encountered.

One of the main problems in CST is that non-expert testers often write vague bug reports that are difficult for developers to understand. A report might describe the problem but may lack further details, such as how to reproduce it, what the expected behavior was, or information about when the bug occurred. If this is the case, then the developer has to invest additional time either to try to understand the report or to ask for further information. In some cases, a potentially useful report may be dropped because of the lack of information. For this reason, the quality of bug reports is an essential concern in CST, as it affects the usefulness of the reports. A common attempt to address this problem was motivating testers through rewards, points, or leaderboards. The assumption was that a motivated tester would write a better report. However, motivation and writing are two different skills. A tester who wants to do their best but does not know how to structure the report will still write a poor report. This study takes a different approach to address this problem: it focuses on supporting the tester rather than motivating them.

A web-based prototype, called CrowdTest, was designed with an interface that provides users with a quest-like prompt before the test and optional hints during reporting. The main aim was to minimize the mental effort of writing a good report by providing a clear, easy-to-follow structure. An experiment was conducted with 30 participants, with dynamic routing to maintain the balance of the two groups. Each bug report was independently rated by three evaluators using a criterion-based rubric. The rest of the paper discusses the design of

CrowdTest, the analysis decisions, and the impact of providing cognitive support for crowdsourced bug reporting. This study was guided by the research question, "Does cognitive scaffolding before testing and during reporting improve the clarity, reproducibility, completeness, and usefulness of a bug report?"

2. Background and Related Work

2.1 The Bug Report Quality Problem

Many bug reports submitted by non-expert testers share a common problem. The bug report may only have a one- or two-line description, such as "the application stopped working," and no additional information. In a case study of more than 27,000 bug reports, Hooimeijer and Weimer [1] found that the quality of the bug reports impacts the time taken to fix bugs. Bug reports that are not clearly explained are more likely to be closed as invalid or not fixed at all. It is not only the content of the report that matters, but also how it is organized. A non-technical user may discover a bug but fail to report it properly. Participants may leave out important details or may feel certain steps were too obvious to be included. In many cases, the bug itself may not have been complex; rather, it might be difficult to express what happened.

2.2 Gamification in Software Testing

Gamification is the application of game design and principles in a non-game context. Deterding et al. [2] describe it as the use of game design elements such as points, badges, and leaderboards in non-game situations. While gamification is often implemented through rewards mechanisms, it also includes non-reward elements such as quest-style framing and structured prompts. This study uses the latter form: non-reward gamification. However, gamification with rewards has not been fully successful in the context of software testing. Even though it increases the fun and engagement, the quality of the bug reports does not necessarily increase. Flatla et al. [3] showed that although gamification enhanced the enjoyment and engagement of the calibration tasks, it did not necessarily improve the quality of bug reports. This suggests that a motivation mechanism alone may be insufficient to improve the quality of structured outputs such as bug reports.

2.3 Cognitive Scaffolding

Cognitive scaffolding refers to providing temporary support to solve a problem that the learner could not solve by themselves. The concept of scaffolding was initially proposed by Wood et al. [4] in the context of educational psychology. Their results showed that tutoring not only provided a solution to the problem but also simplified it, maintained the learner's interest, and helped the learner solve it independently. In crowdsourcing environments, structured tasks have been shown to directly influence the output quality. Kittur et al. [5] showed that redesigning a task to include structured steps significantly improved the output compared to the unguided condition. Similarly, Salehi et al. [6] found that providing structured context to users on complex writing tasks improved the quality of their work. In bug reporting, cognitive scaffolding makes the structure of reporting more transparent to testers. Labeled sections, suggestions, and even optional hints can guide users to the right direction. For example, CUEZILLA [7] supported testers by providing feedback for a completed report and suggesting what information should be included. CrowdTest is built on this idea and provides users with the option of receiving hints and tips throughout the reporting process.

3. System Design

3.1 System Overview

The application, called CrowdTest, was designed, and it managed the full lifecycle of a study from participant consent and conditional routing to bug report submission. A separate extended interface was created to help evaluators review the bug report. Figure 1 illustrates the overall design of the entire system and provides an illustration of how each user moves through the different stages of the process.

Figure 1. System Architecture of CrowdTest

3.2 Staged Architecture

The staged architecture of CrowdTest is summarized in Table 1. The workflow starts with anonymous participation and consent, followed by conditional routing, test application interaction, structured bug reporting and reflection, and finally blinded evaluation of the reports. Originally, the system was designed to log hint clicks, user interaction, and testing duration. However, due to technical issues, some testing durations were stored as 0, preventing reliable analysis on user interaction patterns.

Table 1. CrowdTest Staged Architecture

Stage	Name	Description
1	Setup and Consent	Participants joined the study anonymously, agreed to the consent form, and were assigned to one of two conditions based on a dynamic routing algorithm.
2	Conditional Routing	Gamified participants were redirected to a page with some suggestions of what to do before the test. Baseline participants were redirected to their test system.
3	Testing	All participants interacted with the test system and then proceeded to the bug reporting step.
4	Reporting and Reflection	All testers used a standard reporting form that included fields like title, steps to reproduce, expected behavior, actual behavior, and an option to upload a screenshot. In the gamified condition, each field contained inline, collapsible (by default) hints. After submission, all participants were asked to provide a confidence rating and optional reflection responses.
5	Reports Evaluation	Independent raters blindly graded the bug reports without knowledge of who wrote them or which interface was used to write them.

3.3 Dynamic Routing Algorithm

To ensure an even distribution, the system uses a dynamic routing algorithm after the consent stage. The system queried the database to determine which experimental condition has fewer completed submissions and assigned the incoming participant accordingly. If both conditions had equal counts, one condition was selected at random. Within the selected condition, the participant was then assigned to the scenario with the lowest number of completed submissions. If multiple scenarios had equal counts, one scenario was selected at random. The corrected routing logic is formalized in Algorithm 1 and visualized in Figure 2. As a result of an implementation problem at the beginning of the study, the algorithm incorrectly counted all sessions, including those where users had created a session, but did not submit a report. This resulted in a slight imbalance in the number of sessions per scenario (see Table 5). The issue was fixed immediately but caused a slight imbalance in the observed distribution. The algorithm loads C x S submitted report counts from the database and computes condition totals with the time complexity of O(C x S). With fixed C = 2 conditions and S = 4 scenarios, the table holds only eight entries, so assignment runs in constant time per participant.

Algorithm 1: Dynamic Routing Algorithm

conditions = [baseline, gamified]

scenarios = [A, B, C, D]

count[condition, scenario] = submitted report count

Output: *assigned condition and scenario*

Load count[condition, scenario] for all pairs from the database.

for each *condition do*

total[condition] ← sum of count[condition, s] for all scenarios s

end for

minTotal ← minimum value in total[]

candidates ← conditions where total[condition] = minTotal

if *|candidates| > 1* ***then***

selected_condition ← candidates[floor(rand() × |candidates|)]

else

selected_condition ← candidates[0]

end if

minCount ← minimum of count[selected_condition, s] for all scenarios s

candidates ← scenarios where count[selected_condition, s] = minCount

if *|candidates| > 1* ***then***

selected_scenario ← candidates[floor(rand() × |candidates|)]

else

selected_scenario ← candidates[0]

end if

return *(selected_condition, selected_scenario)*

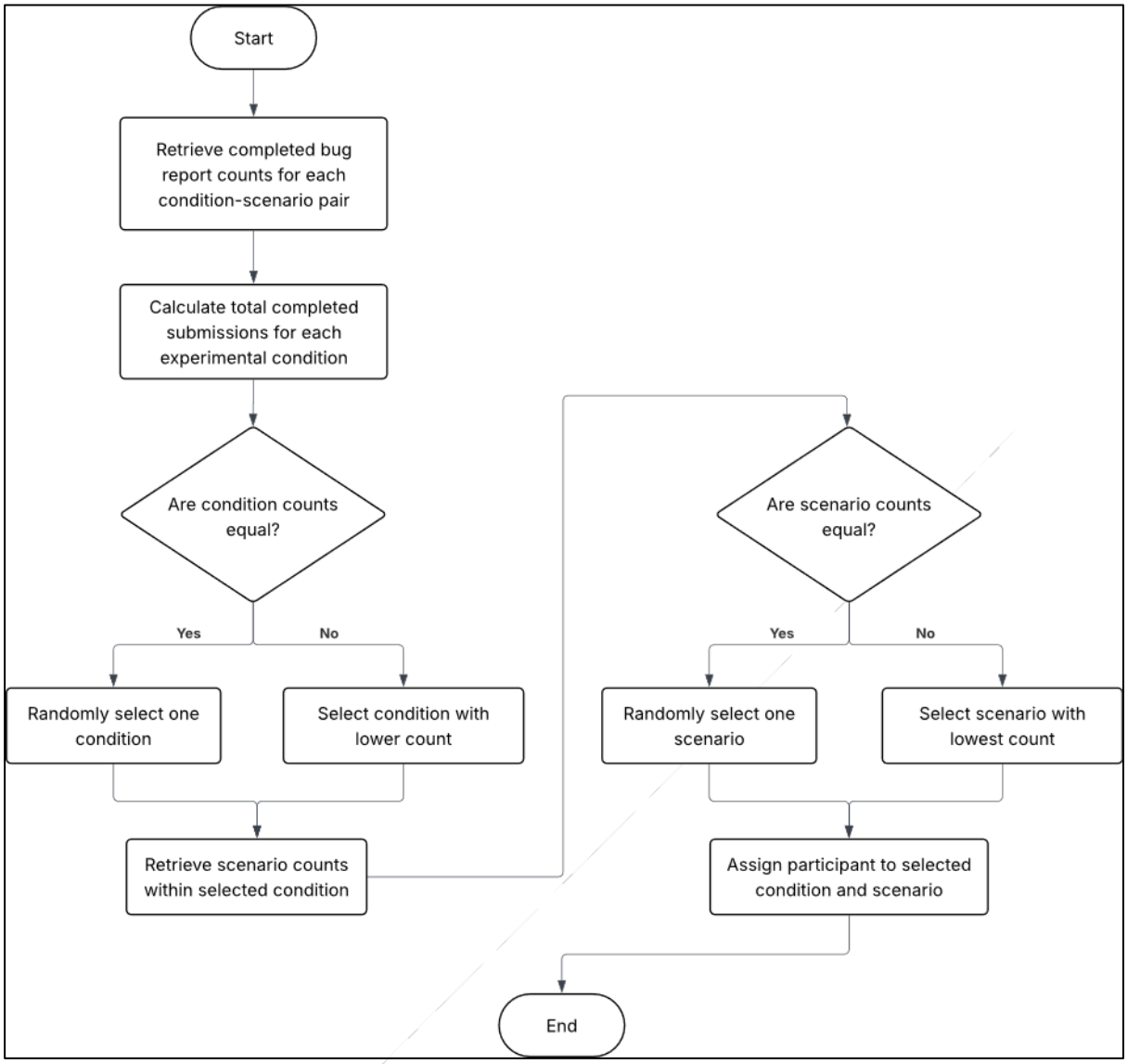

Figure 2. Flowchart of a Dynamic Routing Algorithm for Balancing Participant Assignments

3.4 Cognitive Scaffolding Model

CrowdTest is designed for cognitive support rather than motivational incentives. No points, leaderboards, or any sort of rewards-based system were implemented. In this study, the term “gamified condition” refers to an interface that uses framed presentation elements, such as quest-style prompts and optional inline hints, to provide cognitive support rather than reward-based motivation. In the non-reward gamified condition, participants were shown a list of optional suggestions before interacting with the test application. These suggestions were framed as missions rather than instructions and carried no points or rewards. A representative pre-test suggestion interface is shown in Figure 3.

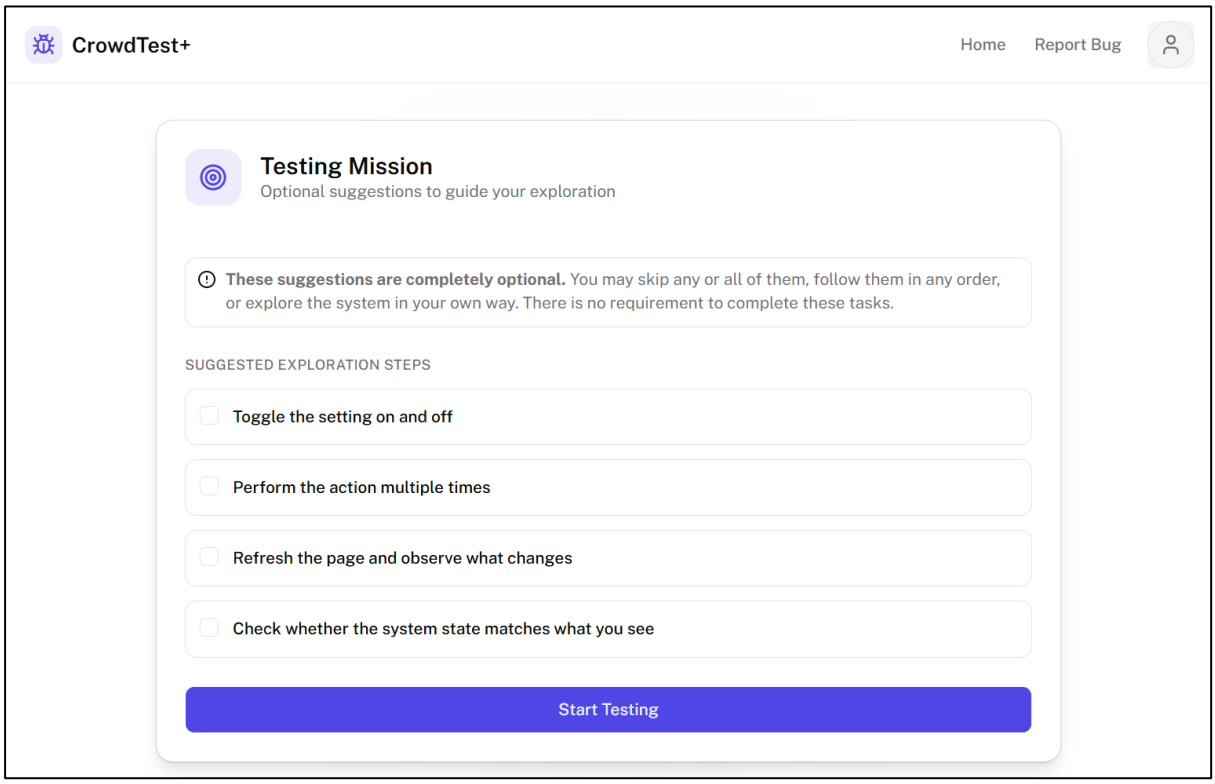

Figure 3. Pre-Test Suggestion Interface Shown to Participants Assigned to the Gamified Condition in Scenario B

The pre-test quest prompts shown to participants in the gamified condition are listed in Table 2, organized by scenario.

Table 2. Pre-Test Quest Prompts by Scenario

Scenario	Quest Prompts
A (Web Form)	Try submitting the form with empty required fields, Enter invalid data and attempt to submit, Correct the data and submit again, and observe any messages or feedback shown
B (State Management)	Toggle the setting on and off, Perform the action multiple times, Refresh the page and observe what changes, Check whether the system state matches what you see
C (Multi-step Workflow)	Move through all steps of the process, Use Back and Next buttons, Enter information on one step and continue, and observe whether your input is preserved.
D (System Feedback)	Perform the main action (login, save, submit), Observe any success or error messages, Repeat the action if possible, Compare what the system says with what actually happens.

The second layer is inline hints during the bug reporting. Each field has a small collapsible hint section. The hints were simple, instructional, and hidden by default. The baseline form shared the same reporting form but without any hints. The cognitive hints available to participants in the gamified condition are listed in Table 3, and reporting interfaces used in both conditions are compared in Figure 4.

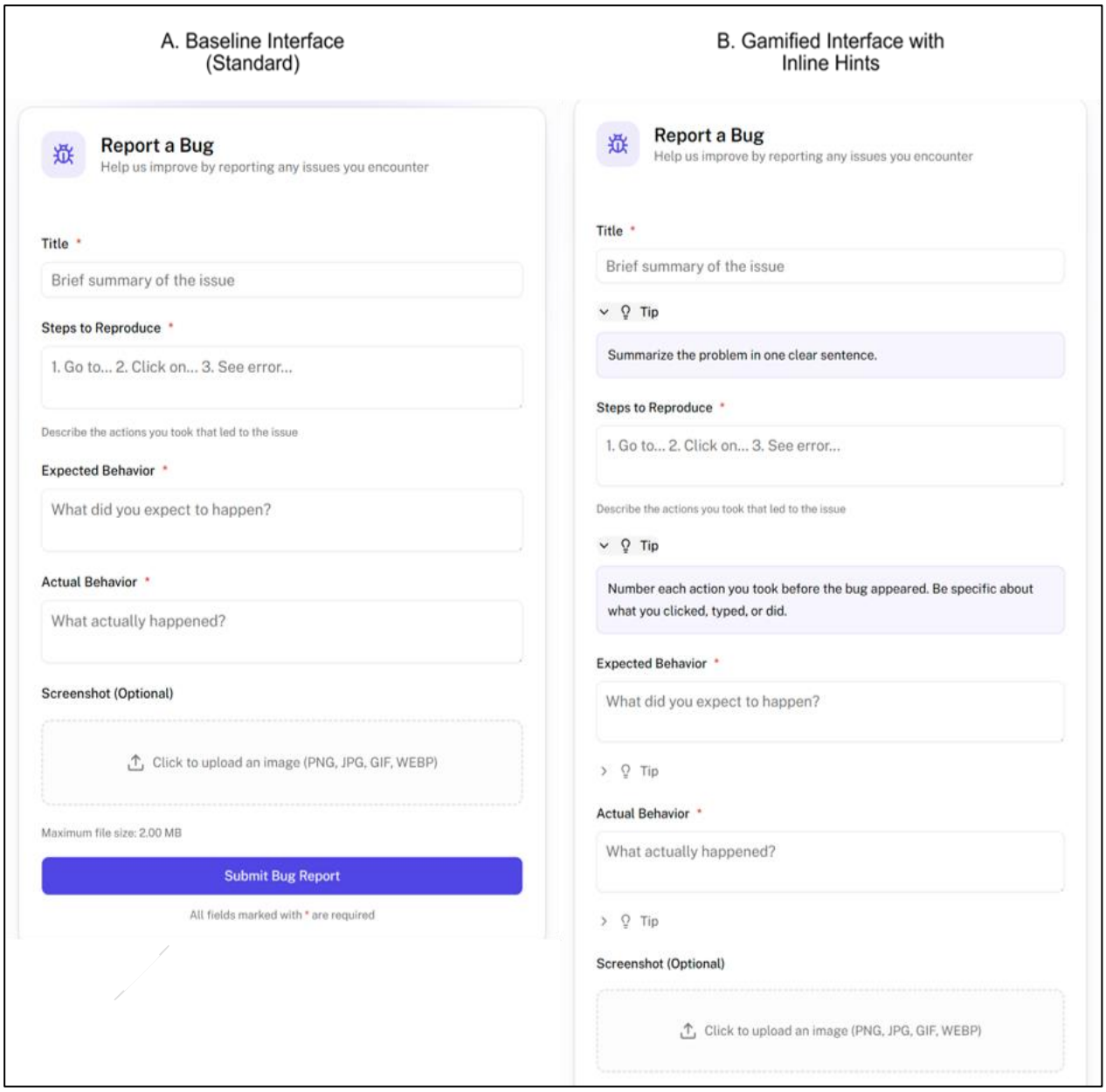

Figure 4. Reporting Interfaces Used in the Study: (A) Baseline Reporting Form and (B) Gamified Reporting Form with Optional Inline Cognitive Hints

Table 3. Cognitive Hints by Form Field

Form Field	Hint Text (Shown on User Request)
Title	Summarize the problem in one clear sentence.
Steps to Reproduce	Number each action you took before the bug appeared. Be specific about what you clicked, typed, or did.
Expected Behavior	Describe what you thought should happen.
Actual Behavior	Describe what actually happened instead.

3.5 Internal Test Application and Seeded Bugs

To create a variable level of difficulty, there were four possible test environments for each tester. Testers were routed to the four-test environment through an adaptive routing method that selects tests from the least complete reports. The system was deliberately seeded with bugs for participants to find. A summary of the bugs that were seeded into the system can be seen in Table 4.

Table 4. Bug Summary by Scenario

Scenario	Component	Bugs	Description
A	User Registration Form	6	Bypass required field, generic error message, 2-second error timeout, stale form state, wrong input type, false success feedback
B	Notification Preferences	3	State not loaded from local storage, 30% chance to corrupt data, false success feedback
C	Account Setup Wizard	3	The progress bar is stuck on back navigation, the name field is cleared when moving from one step to another, name shows (Not provided) during confirmation
D	Application Settings	4	Loading stops early, 30% success toast on failure, 20% error toast on success, conflicting toasts

4. Study Methodology

4.1 Participants

Participation was anonymous and voluntary. Participants were invited through a LinkedIn post and a message on Facebook Messenger. The platform was open to any interested person. No specific age, gender, or detailed technical background data were collected to preserve anonymity. Only session IDs were used for identifying participants throughout the

study. As no demographic or technical background was collected, participant expertise could not be verified.

Only participants who completed the consent form, interacted with the assigned task, and submitted were selected for the study. All other participants were eliminated from the study by the automatic exclusion mechanism of the system. This resulted in a total of 54 participants with 30 valid bug reports, with 15 in the gamified condition and 15 in the baseline condition. Table 5 shows how each participant was classified within Stage 1 of the study. Although there was a slight imbalance between scenarios C and D, the total number of participants per group remained the same.

Table 5. Dynamic Routing Grouping

Scenario	Gamified Condition	Baseline Condition
A	4	4
B	3	3
C	5	4
D	3	4

4.2 Procedure

The flow that each tester went through was the same based on the experiment conditions. After providing consent, the dynamic routing algorithm automatically assigned an experiment mode and scenario ID. The participants used the internal test application to identify bugs. In the gamified condition, participants were shown optional suggestions before the testing application. No time limits were imposed. Before submitting a bug report, all participants were required to acknowledge the instruction “Please describe the bug as clearly as possible so that a developer can reproduce it.” Participants submitted the bug report using a structured form. In the gamified condition, inline hints were available to guide the participants, which were optional and closed by default. After submission, participants were asked to answer a confidence rating question with an optional reflection question. The estimated time for the whole process was around 10–15 minutes.

4.3 Evaluation: Rubric and Evaluators

Three evaluators independently rated all 30 bug reports. Because each report was scored by three evaluators and each participant was assigned to one of four scenarios, the data have a nested structure at the rater and scenario levels. For that reason, the average scores should be interpreted as a simplified analysis, and future studies should use a mixed-effects model to account for these dependencies more fully.

Each evaluator was provided with the bug's reference document containing a description of all 15 seeded bugs and a criterion-based rubric for scoring reports on clarity, reproducibility, completeness, and usefulness. Their agreement was validated through Krippendorff's α before any analysis was conducted. The following instructions were given to evaluators.

- Judge only what is written. Do not assume or fill in missing information.
- Do not penalize for language. Most reports were written in Romanized Nepali. Since the evaluators were native Nepali speakers, they could accurately interpret the context.
- Do not rate based on length. There might be long, vague reports that can get a score of 1, or vice versa.
- Do not penalize for finding limited bugs within the system.
- Ignore spelling and grammar mistakes unless they are difficult to understand.

Evaluators were instructed to judge the report based on four dimensions: clarity, reproducibility, completeness, and usefulness. The rubric scores used for the evaluation are summarized in Table 6.

Evaluations were conducted under blind conditions. The evaluators were provided with an interface that displayed only the bug report content (title, steps, expected and actual behavior, and optional screenshot). No experiment mode, quest usage, hint usage, or participant confidence ratings were disclosed to the evaluator to avoid biases. Once completed, the evaluators exported the evaluation JSON and forwarded it to the researcher.

Table 6. Bug Report Evaluation Rubric

Dimension	Evaluator Question	Scale Anchors
Clarity	Is the report easy to understand?	1 = Difficult to understand 5 = Fully understandable, no guessing needed
Reproducibility	Could a developer follow these steps and see the same bug?	1 = No steps or completely useless 5 = Will reproduce bug every time
Completeness	Did the tester fill in everything properly?	1 = Missing most information 5 = All present, no gaps
Usefulness	Would this report actually help a developer fix the bug?	1 = No actionable information 5 = Fully actionable; the developer can start immediately.

5. Analysis

5.1 Inter-Rater Reliability

Before analyzing the data, it was necessary to verify inter-rater agreement. For example, if every reviewer consistently disagreed with every other reviewer, then taking the average of their ratings may provide a misleading result. Krippendorff's α was used to assess agreement among reviewers using the ordinal distance metric [8]. Krippendorff's α was preferred over Cohen's kappa because it can work with three or more raters, whereas Cohen's kappa requires pair-by-pair calculations. Also, the ordinal distance metric considers a difference between ratings of 1 and 5 to be greater than a difference between ratings of 3 and 4, which is appropriate for this experiment as the difference between rating levels cannot be considered to be equivalent.

Following Krippendorff's α rules [8], $\alpha \geq 0.800$ denotes solid agreement, $\alpha \geq 0.667$ represents sufficient agreement so that tentative conclusions may be made, and $\alpha < 0.667$ is indicative of unreliable agreement. The calculated α values for all dimensions are presented in Figure 5. The overall score is the unweighted average of the four-dimension scores.

Clarity, reproducibility, usefulness, and the overall composite score ($\alpha = 0.748$) achieved at least provisional agreement, whereas completeness fell below the recommended threshold. As a result, completeness is presented for observational purposes, and any overall composite score should be interpreted cautiously because it includes this weaker dimension.

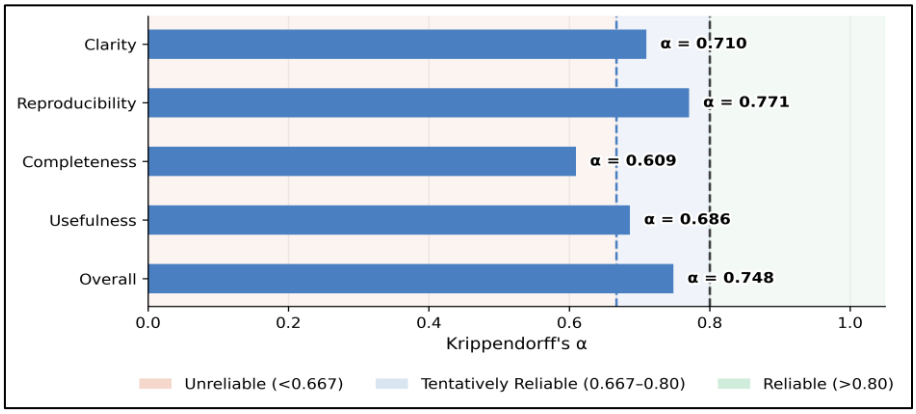

Figure 5. Krippendorff's α Scores Showing the Reliability of Evaluator Agreement for Each Dimension

5.2 Normality Testing

The Shapiro-Wilk test was used to determine whether the scores in each dimension were distributed normally. This test is recommended over alternatives like Kolmogorov-Smirnov for small samples ($n < 50$) because it has more statistical power to correctly reject the null hypothesis of normality [9]. The value of $p > 0.05$ indicates that data are distributed normally.

Table 7. Shapiro-Wilk Test Statistics

Dimension	Gamified W	Gamified p	Baseline W	Baseline p
Clarity	0.6298	< 0.0001	0.7991	0.0036
Reproducibility	0.7742	0.0017	0.8957	0.0820
Completeness	0.5900	< 0.0001	0.7914	0.0029
Usefulness	0.6778	0.0001	0.8666	0.0301
Overall	0.6718	0.0001	0.8183	0.0064

The results of the Shapiro-Wilk test indicate that scores across nearly all dimensions significantly deviated from the normal distribution ($p < 0.05$), except for the reproducibility of the baseline group ($p = 0.082$). These patterns are visualized in Figure 6 as Q-Q plots. Because nearly all of the data failed to meet assumptions of normality, a Mann-Whitney U test was chosen to compare the performance between the gamified and baseline conditions. The full Shapiro-Wilk test statistics (W and p-values) for each dimension are listed in Table 7.

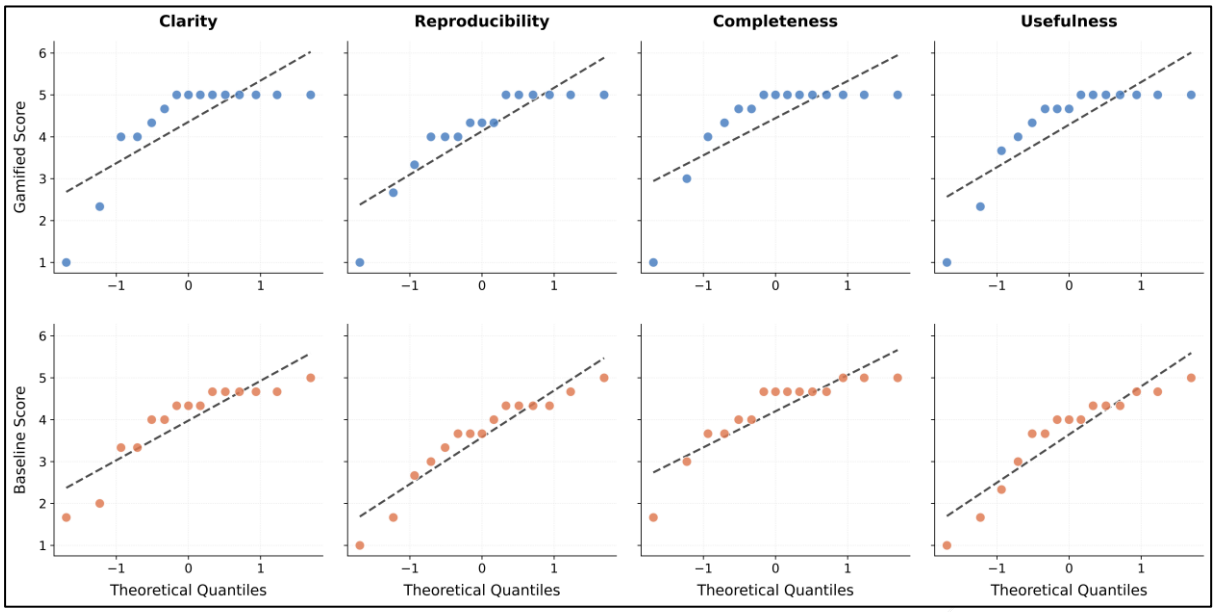

Figure 6. Q-Q Plots Evaluating the Normality Across Four Primary Dimensions

5.3 Descriptive Statistics

Table 8 shows the mean, standard deviation, and median for each dimension, separated by experimental condition. Because the data were non-normal, the medians are the primary descriptive summary. In every dimension, the gamified condition scored higher than the baseline group. 95% bootstrap confidence intervals for each mean difference were computed using 2,000 resamples (seed = 42) and are reported in Table 8a.

Table 8. Descriptive Statistics by Condition

Dimension	G Mean	G SD	G Median	B Mean	B SD	B Median	ΔM
Clarity	4.356	1.178	5.000	3.978	0.996	4.333	+0.378
Reproducibility	4.133	1.111	4.333	3.578	1.109	3.667	+0.556
Completeness	4.444	1.103	5.000	4.200	0.915	4.667	+0.244
Usefulness	4.289	1.167	4.667	3.644	1.158	4.000	+0.644
Overall	4.306	1.124	4.750	3.850	1.015	4.250	+0.456

G = Gamified, B = Baseline, ΔM = Gamified mean - Baseline mean, and SD = Standard deviation.

All confidence intervals include zero because bootstrap CIs computed on raw mean differences are sensitive to variance at this sample size. This does not contradict the Mann-Whitney results in Table 9.

Table 8a. Bootstrap CIs for Mean Differences

Dimension	ΔM	95% CI Lower	95% CI Upper
Clarity	+0.378	- 0.400	+1.089
Reproducibility	+0.556	- 0.222	+1.289
Completeness	+0.244	- 0.511	+0.911
Usefulness	+0.644	- 0.156	+1.400
Overall	+0.456	- 0.289	+1.156

ΔM = Gamified mean - Baseline mean, and CI = Confidence intervals

Rank-based effect size intervals in Figure 9 are the primary measure of uncertainty for this data. The relative performance of the two conditions across all dimensions is summarized in the radar chart in Figure 7.

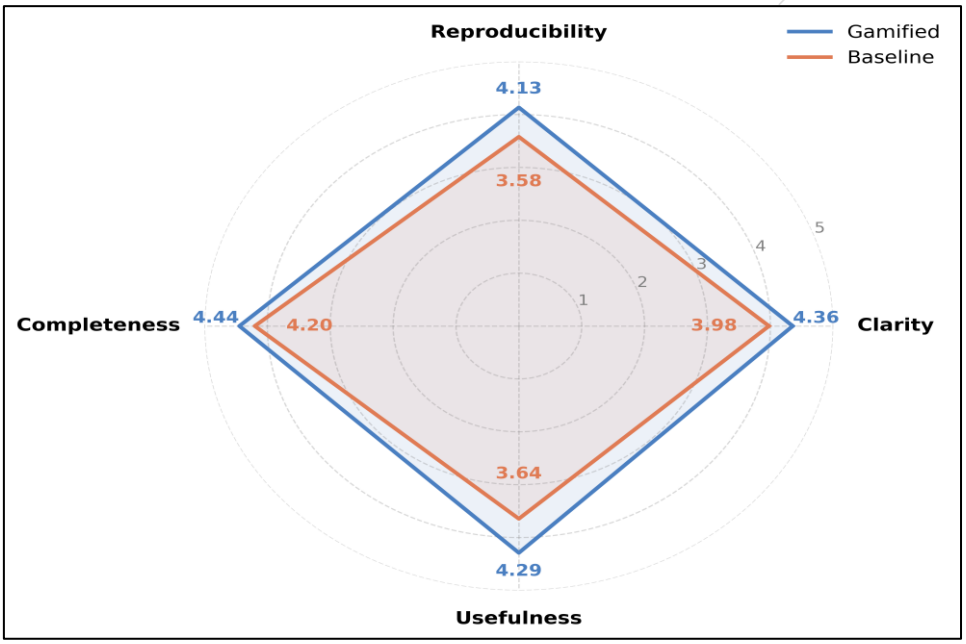

Figure 7. Quality Radar Chart of Each Dimension in Gamified and Baseline Conditions

5.4 Mann-Whitney U Tests

The Mann-Whitney U test [10] was performed to determine if there were statistically significant differences between experimental conditions. Two-sided tests were used throughout the process to test in both directions rather than assuming that gamification could only improve scores. The rank-biserial correlation (r) derived from the U-statistic was used as the effect size measure. Following Cohen's standard convention [11], $|r| < 0.10$ was interpreted as negligible, 0.10–0.30 as small, 0.30–0.50 as medium, and >0.50 as large. Table 9 presents the results.

Table 9. Mann-Whitney U Test Results

Dimension	U	p	r	Effect Size
Clarity	161.5	0.039	0.436	Medium
Reproducibility	153.0	0.092	0.360	Medium
Completeness	152.0	0.091	0.351	Medium
Usefulness	166.5	0.024	0.480	Medium
Overall	165.5	0.029	0.471	Medium

U = Mann-Whitney U statistic, p = probability, r = rank-biserial correlation

The gamified condition scored significantly higher than the baseline group on clarity, usefulness, and the overall composite score, each with a medium effect size. While reproducibility also favored the gamified condition, the difference was not significant, and completeness is reported for observational purposes only because of its low IRR. The overall composite score is the unweighted mean of all four dimensions, including completeness, so its result should be interpreted with caution and should not be treated as the primary finding of this study. The distributions of evaluation scores are visually represented in the box plots in Figure 8.

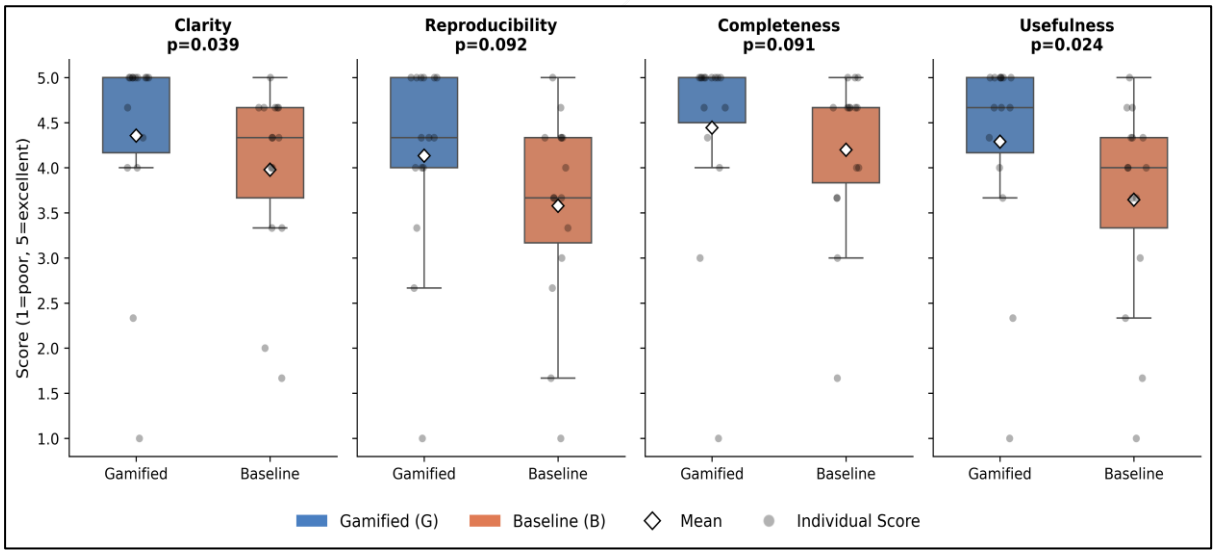

Figure 8. Box Plot Representing Score Distribution by Dimension

5.5 Bootstrap Confidence Intervals

95% bootstrap confidence intervals [12] were computed for each effect size using 2,000 resamples at a fixed random seed (42). This approach was chosen because standard methods for confidence intervals assume normally distributed data, which was not the case here (see Section 5.2).

The resulting rank-biserial correlation effect sizes and their corresponding 95% confidence intervals are visualized in the forest plot in Figure 9. All five effect sizes were positive. The confidence intervals for reproducibility and completeness include zero, which is consistent with their non-significant p-values, and the intervals for clarity and usefulness are above zero, consistent with their significant results. All intervals are wide because of the small sample size, which is expected given n=15 per group, and a larger study would produce more precise estimates.

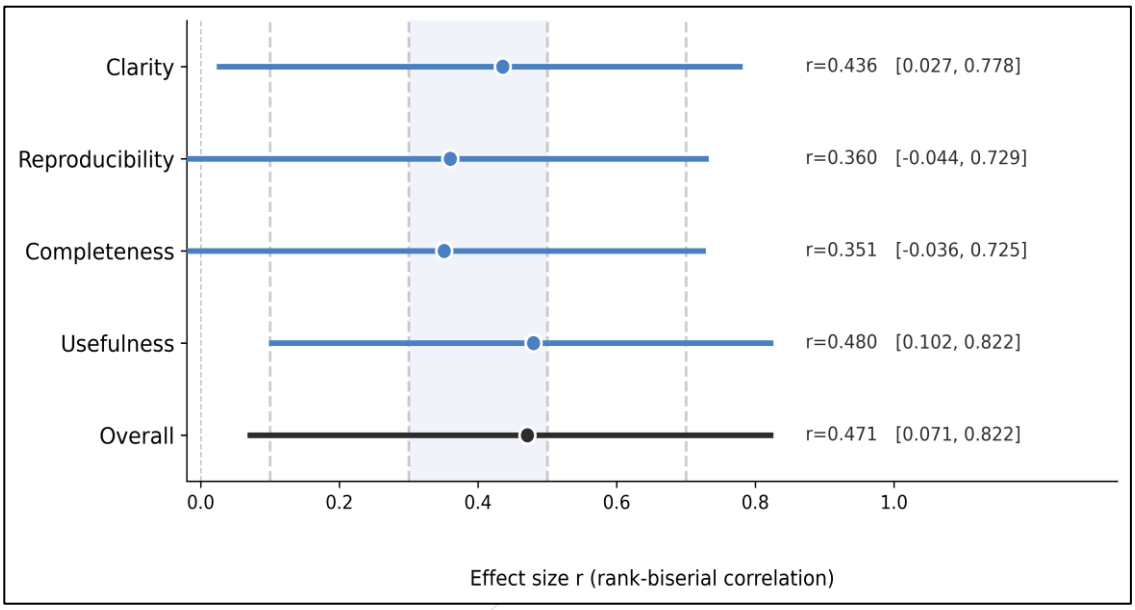

Figure 9. Forest Plot of Effect Sizes with 95% Bootstrap CI (2,000 resamples)

5.6 Post-Hoc Power Analysis

The non-significant results of reproducibility and completeness do not mean that there is no effect. A p-value above 0.05 can also mean that the sample size was insufficient to detect the effect. To measure this, a post-hoc power analysis was done.

The rank-biserial correlation r was converted to Cohen's d using the formula $d = \frac{2r}{\sqrt{(1-r^2)}}$ to make values compatible with the statsmodel's power function, which requires Cohen's d as input. Power was calculated assuming α = 0.05 and a two-tailed test. Table 10 presents the estimated number of participants to reach 80% power.

Table 10. Post-Hoc Power Analysis

Dimension	r	d	Need n/group	Need N total	Our Power (%)	Status
Clarity	0.436	0.968	18	36	72.5	Needed 3 more
Reproducibility	0.360	0.772	28	56	53.2	Needed 13 more
Completeness	0.351	0.750	29	58	50.9	Needed 14 more
Usefulness	0.480	1.094	15	30	82.5	Enough
Overall	0.471	1.068	15	30	80.6	Enough

80% power target, $\alpha = 0.05$, two-tailed; actual study: n = 15 per group (N = 30 total)

Based on Table 10, reproducibility and completeness would require approximately 28-29 participants per group to reach the target power of 80%. This suggests that insufficient sample size may be the primary explanation for the non-significant results. Therefore, their results should be interpreted as underpowered rather than as evidence of no effect. Based on statistical calculations, reaching 80% power requires:

- 44 total participants (22 per group) for a medium-large effect (r = 0.40)
- 82 total participants (41 per group) for a medium effect (r = 0.30)
- 120 total participants (60 per group) for a small-medium effect (r = 0.25)

6. Discussion

6.1 Interpretation of Results

The results indicate that the gamified condition achieved higher average scores across all measured dimensions, with statistically significant differences for clarity, usefulness, and overall composite score. There were no reward systems in the gamified condition: no point, no badge, and no leaderboards were integrated into the system. The only additions were optional quest-like prompts before testing and collapsible inline hints during reporting. This suggests that cognitive scaffolding may help participants produce clearer and more useful bug reports. Among the four dimensions, usefulness showed the largest mean difference ($\Delta M = +0.644$, r = 0.480). One possible explanation is that the quest prompts and inline hints helped participants to explicitly describe expected and actual behavior that occurred. On the other hand, a collapsible hint alone may not be enough for reproducibility. This suggests that future versions should include more targeted prompts specifically for reproduction steps.

6.2 The Completeness Problem

Among the four dimensions, completeness produced the lowest inter-rater reliability ($\alpha = 0.609$), which is below the standard 0.667 threshold. This suggests that evaluators did not interpret the completeness rubric as precisely as clarity, reproducibility, and usefulness. The most likely reason is that the rubric for completeness was not specific enough, and evaluators may have interpreted it differently. For this reason, completeness should be interpreted cautiously and solely presented for observation purposes only. Since the overall composite score includes completeness, it should be interpreted with caution. Future studies should refine this dimension and provide clearer example reports for each score level before drawing conclusions from this dimension.

6.3 Limitations

This study has several limitations that affect the strength of its conclusions. First, the sample size was small, with only 30 valid reports. Although it allowed detection of a medium effect size, it limits statistical power for reproducibility and completeness. Second, the participants were completely anonymous, and no technical data were collected, so the study cannot verify the expertise level of each participant. Participants were assigned to different systems, each having different bugs, so variation in difficulty may have influenced the bug report quality instead of cognitive support. Although dynamic routing balanced group sizes across scenarios, it did not control for scenario difficulty. In addition, all evaluators rated a full set of 30 reports across four different scenarios, so evaluator fatigue is a real concern. Before running the group comparisons, the three rater scores were averaged. This approach does not model how much rater or scenario-level differences may have affected the results. Moreover, technical issues prevented reliable collection of telemetry data such as testing duration. As a result, the study cannot directly explain how participants interacted with the support features during the task.

7. Conclusion

The study introduced CrowdTest, designed to help participants before testing and during bug reporting through quest-like prompts and optional inline hints. In this pilot experiment, the gamified condition performed significantly better compared to the baseline on clarity ($p=0.039$), usefulness ($p=0.024$), and overall composite score ($p=0.029$), with medium

effect sizes. Reproducibility showed positive but non-significant growth ($p=0.092$), while completeness was reported only for observational purposes due to its low inter-rater agreement ($\alpha=0.609$). Post-hoc analysis suggested that the sample used for the study was probably inadequate to detect statistically significant differences for those two dimensions. These findings should be interpreted cautiously because of the low sample size, scenario variation, non-expert evaluators, and missing telemetry data. Participants were anonymous volunteers from unknown backgrounds who participated via social media, which restricts the external validity to similar non-expert groups. Even with these limitations, the study provides preliminary evidence that gamified cognitive support may be a useful design direction for improving the bug report quality in crowdsourced testing environments. Future studies should include a larger sample, with approximately 28-29 participants per group as indicated by power analysis, and ensure that scenario difficulty is balanced across conditions. Reliable telemetry data collection would also improve the reliability of the findings.

References

[1] Hooimeijer, Pieter, and Westley Weimer. “Modeling Bug Report Quality.” Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering (New York, NY, USA), ASE ’07, November 5, 2007, 34–43.

[2] Deterding, Sebastian, Dan Dixon, Rilla Khaled, and Lennart Nacke. “From Game Design Elements to Gamefulness: Defining ‘Gamification.’” Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (New York, NY, USA), MindTrek ’11, September 28, 2011, 9–15.

[3] Flatla, David R., Carl Gutwin, Lennart E. Nacke, Scott Bateman, and Regan L. Mandryk. “Calibration Games: Making Calibration Tasks Enjoyable by Adding Motivating Game Elements.” Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (New York, NY, USA), UIST ’11, October 16, 2011, 403–12.

[4] Wood, David, Jerome S. Bruner, and Gail Ross. “The Role of Tutoring in Problem Solving.” Journal of Child Psychology and Psychiatry 17, no. 2 (1976): 89–100.

[5] Kittur, Aniket, Ed H. Chi, and Bongwon Suh. “Crowdsourcing User Studies with Mechanical Turk.” Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (New York, NY, USA), CHI ’08, April 6, 2008, 453–56.

[6] Salehi, Niloufar, Jaime Teevan, Shamsi Iqbal, and Ece Kamar. “Communicating Context to the Crowd for Complex Writing Tasks.” Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (New York, NY, USA), CSCW ’17, February 25, 2017, 1890–901.

[7] Bettenburg, Nicolas, Sascha Just, Adrian Schröter, Cathrin Weiss, Rahul Premraj, and Thomas Zimmermann. “What Makes a Good Bug Report?” Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (New York, NY, USA), SIGSOFT ’08/FSE-16, November 9, 2008, 308–18.

[8] Krippendorff, Klaus. Content Analysis: An Introduction to Its Methodology. Fourth Edition, Thousand Oaks, CA: SAGE Publications, Inc., 2019, 403-407

[9] Shapiro, S. S., and M. B. Wilk. “An Analysis of Variance Test for Normality (Complete Samples).” Biometrika 52, no. 3/4 (1965): 591–611.

[10] Hinton, Perry R. "Mann–Whitney U Test." In Encyclopedia of Research Design, edited by Neil J. Salkind. Thousand Oaks, CA: SAGE Publications, Inc., 2010, 747-50.

[11] Cohen, Jacob. “Statistical Power Analysis for the Behavioral Sciences.” 2nd ed. Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

[12] Efron, Bradley, and R. J. Tibshirani. An Introduction to the Bootstrap. New York: Chapman and Hall/CRC, 1994.