# Test Royale: A Game-Based Approach for Measuring Unit Test Quality

Menna Alaa Aldeen
menna.rasheed@student.giu-uni.de
German International University
Cairo, Egypt

Slim Abdennadher
slim.abdennadher@giu-uni.de
German International University
Cairo, Egypt

Caroline Sabty
caroline.sabty@giu-uni.de
German International University
Cairo, Egypt

Alia Elbolock
alia.elbolock@aucegypt.edu
German International University
Cairo, Egypt

## Abstract

Writing software tests is a challenging and often unpopular task among developers, who may rush the process due to its repetitive nature. In programming education, testing is frequently overlooked, leading students to produce weak and incomplete test cases. Effective unit testing still depends on understanding program context and expected behavior, which cannot be replaced by automated tools. Prior studies indicate that gamification can increase motivation, but its impact on test case quality remains underexplored. The goal of this study is to evaluate whether competitive gamification can enhance test case quality and overall software quality. In this work, we introduce Test Royale, a competitive gamified system that encourages developers to write stronger test cases and produce higher-quality code by leveraging competition between players.

## Keywords

Software Testing, Gamification, Unit Testing, Test Quality, Test Coverage

**ACM Reference Format:**
Menna Alaa Aldeen, Slim Abdennadher, Caroline Sabty, and Alia Elbolock. 2025. Test Royale: A Game-Based Approach for Measuring Unit Test Quality. In . ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 Introduction

White-box testing is an essential approach in software testing, relying on the developer's knowledge of the internal logic of the software. Its main goal is to verify the functionality and correctness of the code, ensuring it meets the desired behaviour. One of the core elements of white-box testing is unit testing, which focuses on validating each part of the code individually [10]. This practice is maintained throughout the development process to ensure software reliability [8]. Unit testing is important because it allows testers to access internal program details, enabling them to quickly identify logical errors and complex problems. It also enhances collaboration between testers and developers, helping define issues more precisely. Furthermore, unit testing assists in identifying security vulnerabilities and performance issues by analysing specific code sections and testing inputs that may cause exceptions [10].

Despite its importance, software testing education remains inadequate, and teaching the necessary skills continues to be a challenge [7]. Keeping students engaged while learning software testing concepts is difficult. Similarly, in industry, testing is often considered a boring task and is sometimes neglected by developers, which can lead to software failures—even in critical systems such as banking, aviation, and automotive applications, which can lead to death [6]. While automation can help, it is limited, as the effectiveness of a test will always depend on human intuition and understanding of the program [4] [6].

Gamification has been introduced as a potential solution. It is mostly applied in unit-testing tools, showing improvements in user experience and engagement. However, measuring its impact on test quality is challenging [5].Although many studies find that gamification boosts motivation and engagement, its impact on the quality of test cases is still uncertain. Much of the existing work measures activity levels—such as the number of tests written—rather than the correctness or coverage of those tests. As a result, while gamification appears to encourage participation, it is less clear whether it leads to better test-case quality, and some studies question whether it has any real effect on producing higher-quality tests or improving path coverage [2] [9] [1].

In this study, we aim to provide developers with meaningful feedback on the quality of the test cases they write. By combining gamification with clear, measurable metrics, we intend to maintain high motivation and engagement while ensuring that the tests produced are of high quality. This study introduces Test Royale, a competitive gamified system that provides developers with feedback on the quality of the test cases they write, aiming to improve both engagement and test effectiveness


*Conference'17, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 2 Literature Review

Gamification has gained attention as a way to improve motivation, engagement, performance, and efficiency in non-game contexts [5]. Common elements include points, badges, achievements, scoring systems, leaderboards, and challenges [4]. Prior work shows that gamification has already been applied to software testing. A recent review examined existing mechanics, tools, benefits, and challenges. It found that most approaches focus on white-box unit testing, often using Java with JUnit, and are mainly evaluated in educational settings. The review also highlighted that studies use different measurement criteria and called for unified metrics to assess effectiveness and efficiency [5].

Work in software testing education shows similar patterns. When comparing a traditional course with a gamified version, they found that students still learned better through the traditional approach, but gamification increase attention and make the course more engaging [7]. Another study evaluated the IntelliGame tool to examine whether gamification improves engagement, motivation, testing behaviour, and test quality. Testers wrote more test cases and used debugging features more often, but there was no improvement in coverage or mutation score, and the test suites contained more test smells—suggesting a focus on quantity rather than quality [11]. A similar study on developer motivation and unit-test quality reported that the gamified group was more motivated and found more bugs, but path coverage did not improve. Badges were viewed as motivating, and survey feedback was positive [8]. Another study used a gamified exploratory-testing framework themed around Pirates of the Caribbean. It followed a treasure-hunt style and was designed using the Octalysis Gamification Framework. The aim was to improve engagement, motivation, and learning during exploratory testing. The results showed very high engagement ,motivation and satisfaction by students. They were also able to find defects in usability, functionality, and the interface. [3] A study used GoRace, a gamified Olympic-style tool integrated with SQL-Test, to test whether gamification improves student performance and engagement in software testing. The results showed that the gamified group performed better in several exercises, with higher effectiveness, scores and engagement. However, the study did not measure code quality directly. Performance was based only on task effectiveness, not on deeper metrics like coverage or mutation. This means the study shows better performance, but it does not show whether gamification improves test-case quality [2].

Across these studies, gamification consistently increases motivation and engagement. However, its effect on the quality of produced test cases remains unclear, and several studies show that it may lead testers to prioritise scoring and rewards over producing deeper, higher-quality tests.

## 3 Study Aim and Research Question

Previous studies show that gamification consistently increases motivation and engagement in software testing. However, its effect on test-case quality is still unclear. Many games lead testers to write more tests but not necessarily better ones. Because of this gap, our aim is to measure whether gamification can improve code quality, not only motivation. Our research question is:
RQ: Does gamification improve the quality of test cases produced by developers?

## 4 Test Royale Game

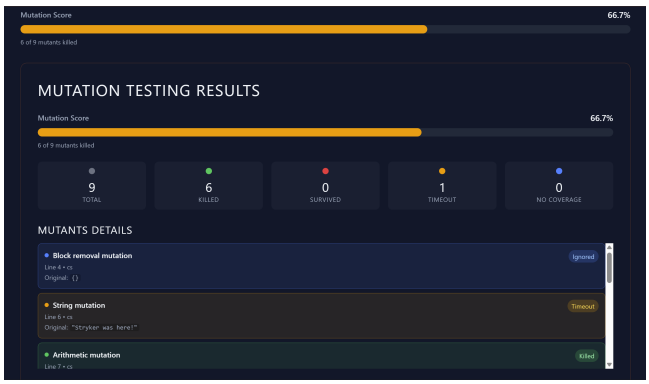


**Figure 1: Detailed view of the mutation score, showing the number of mutants killed, survived, timed out, or not covered for each test case.**

To address our research question, we introduce Test Royale, a competitive multiplayer online game that provides players with direct feedback on the quality of their tests within each round. The game is designed to help players learn, evaluate, and improve their testing skills through repeated play.

### 4.1 Game Flow

When the game starts, each player can either register or log in using their verified email. This allows the system to track their progress over time. After logging in, the player can create a room or join an existing one. A room is created by one player, who becomes the host. When the host creates the room, the system generates a unique room code. The host shares this code with the other players so they can join in his room and start a game together. The host is also the one who decides when to start the match. The main purpose of the room is to connect all players online so they can play in the same session.

When the host starts the game, all players receive the same base code written in C#. along with a countdown timer. Each player is then required to write test cases for that code individually using the MSTest Framework. While playing, players can monitor their test quality using several metrics, which are generated using tools such as Coverlet for coverage measurements and Stryker.NET for mutation testing: lines of code covered percentage, branch coverage percentage, line coverage, execution time, and mutation score. The mutation score is detailed, showing how many mutants were killed, survived, timed out, or not covered as shown in figure 1. Players can also track exactly which lines of the base code they have covered, generated by Coverlet. When the timer ends, the game navigates to the results page. Here, each player's final score is calculated using the following formula:

$$\begin{aligned}\text{Final Score} =& (MutationScore \times 0.4) + (Coverage \times 0.2) \\ &+ (LineCoverage \times 0.1) + (UsefulLOC \times 0.2) \\ &- (ExecutionTime \times 0.1) \end{aligned} \quad (1)$$

In result page players are ranked based on their final scores calculated bu formula, and players with higher scores indicate better performance. The leaderboard shows the ranking for all of the players who joined the same session. In Addition the Results page displays any badges or achievements earned during the match by any player. Each player can view these results alongside the performance of the other participants.

Moreover, every player has a personal profile where they can track their progress over time. The profile shows recent games, average score, win rate, and a summary of their testing performance. This allows players to review how they are improving from one game to another.

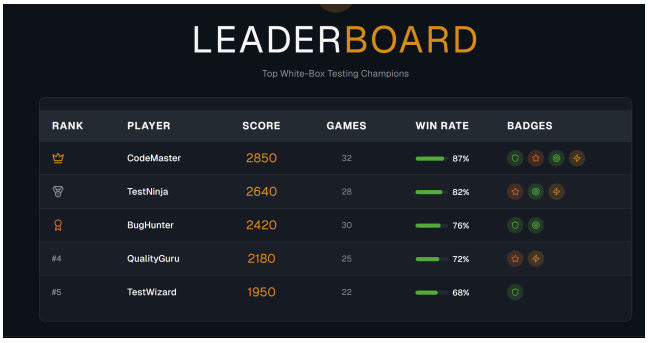


**Figure 2: Overall leaderboard showing rankings from all previously played games, promoting ongoing competition and social comparison.**

## 4.2 Gamification Elements

Test Royale includes several gamification elements aimed at increasing motivation and creating an engaging competitive environment. Other than the scoring system that is calculated during each round, there are badges awarded for specific achievements. For example, the *Mutation Slayer* badge is earned for killing 80% of mutants, while the *Coverage Explorer* badge has tiered levels—Bronze for 70% coverage, Silver for 80%, Gold for 90%, and Platinum for full 100% coverage. Other badges, such as *Lightning Tester* and *Clean Coder*, reward speed and clean, efficient test suites, while *Consistency Champ* recognises players who improve across multiple rounds. Additionally, repeated top performance is acknowledged with a series of *Top Rank* badges, from *Rookie* for the first top-tier achievement up to *Legend* for sustained excellence.

There is also another leaderboard that shows overall rankings from all previously played games, as shown in Figure 2, in addition to the session-specific leaderboard generated at the end of each game. This promotes ongoing competition and social comparison. These design choices are guided by the Octalysis framework, targeting core drives such as achievement through badges, social influence via leaderboards, and a sense of accomplishment through progressive goals. Together, these elements aim to make the testing activity more engaging while motivating players to produce higher-quality test cases.

# 5 Methodology

## 5.1 Study Design

To see how gamification affects testing skills, we ran a study with a simple before-and-after design. First, participants tackled a testing task without any gamified elements, giving us a clear picture of their baseline performance for example test-case quality, coverage, and efficiency. Next, they played Test Royale, where they did a similar task in a gamified setting, complete with points, badges, and a leaderboard. While they played, we tracked their performance and similar metrics inside the game to understand how they performed . After finishing, participants answered a short survey about their experience, motivation, and engagement. By comparing the results from the non-gamified and gamified sessions, we could see whether gamification actually improved the quality and speed of their testing. We looked at metrics like mutation score, line and branch coverage, useful lines of code, and execution time, alongside game-specific indicators such as badges earned and leaderboard ranking, to capture both skill improvement and motivation.

## 5.2 Participants

## 5.3 Experimental Design / Procedure

# 6 Results / Findings

# 7 Discussion

# 8 Limitations

# 9 Conclusion

# References

[1] Davíð Arnarrson and Ívar Húni Jóhannesson. 2015. *Improving Unit Testing Practices With the Use of Gamification.* Master's thesis. Chalmers University of Technology.

[2] Raquel Blanco, Manuel Trinidad, María José Suárez-Cabal, Alejandro Calderón, Mercedes Ruiz, and Javier Tuya. 2023. Can gamification help in software testing education? Findings from an empirical study. *Journal of Systems and Software* 200 (2023), 111647. doi:10.1016/j.jss.2023.111647

[3] Igor Ernesto Ferreira Costa and Sandro Ronaldo Bezerra Oliveira. 2020. The use of gamification to support the teaching-learning of software exploratory testing: an experience report based on the application of a framework. In *2020 IEEE Frontiers in Education Conference (FIE).* 1–9. doi:10.1109/FIE44824.2020.9273943

[4] Gordon Fraser. 2017. Gamification of software testing. In *2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST).* IEEE, 2–7.

[5] Tommaso Fulcini, Riccardo Coppola, Luca Ardito, and Marco Torchiano. 2023. A review on tools, mechanics, benefits, and challenges of gamified software testing. *Comput. Surveys* 55, 14s (2023), 1–37.

[6] Laura Inozemtseva and Reid Holmes. 2014. Coverage is not strongly correlated with test suite effectiveness. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014).* Association for Computing Machinery, New York, NY, USA, 435–445. doi:10.1145/2568225.2568271

[7] Gabriela Martins de Jesus, Fabiano Cutigi Ferrari, Leo Natan Paschoal, Simone do Rocio Senger de Souza, Daniel de Paula Porto, and Vinicius Humberto Serapilha Durelli. 2020. Is It Worth Using Gamification on Software Testing Education? An Extended Experience Report in the Context of Undergraduate Students. *Journal of Software Engineering Research and Development* 8 (Aug. 2020), 6:1 – 6:19. doi:10.5753/jserd.2020.738

[8] M. A. Mascheroni and E. Irrazábal. 2018. Continuous testing and solutions for testing problems in continuous delivery: A systematic literature review. *Computación y Sistemas* 22, 3 (2018), 1009–1038. doi:10.13053/CyS-22-3-2794

[9] Shafiq Saloum and Fredrik Rissanen. 2019. The impact of gamification in unit testing. (2019).
[10] Ján Skalka and Martin Drlík. 2023. Development of automatic source code evaluation tests using grey-box methods: A programming education case study. *IEEE Access* 11 (2023), 106772–106792.
[11] Philipp Straubinger, Tommaso Fulcini, Giacomo Garaccione, Luca Ardito, and Gordon Fraser. 2025. Gamifying Testing in IntelliJ: A Replicability Study. *Proceedings of the ACM on Software Engineering* 2, ISSTA (2025), 2407–2429.