|  |
| --- |
|  |

**1. Introduction**

The Introduction section of the Bug Squasher Test Plan serves as an initial overview, providing essential details about the project and the document. Here’s an explanation of each component:

**Project Name: Bug Squasher - A Gamified Bug Reporting Tool**

This is the name of the project, indicating that it is a tool designed to make bug reporting engaging through gamification elements such as points and badges.

**Project Description**

This part briefly describes what Bug Squasher is and its purpose. Specifically, Bug Squasher is a web-based application designed to make bug reporting fun and engaging for software development teams. It gamifies the process by awarding points and badges to users who submit high-quality bug reports. This description highlights the primary goal of the project and its key feature: gamifying bug reporting to improve engagement and quality.

The introduction section sets the stage for the rest of the test plan. It provides a quick summary of what the project is about, who is responsible for the document, and when it was created. This context is crucial for anyone who might be reading the document, whether they are stakeholders, team members, or external auditors, as it gives them a clear starting point and understanding of the document’s relevance.

**Test Objectives**

The test objectives for Bug Squasher encompass ensuring the seamless functionality of bug reporting features, validation of severity level assignment, confirmation of the point and badge system's accuracy, verification of the leaderboard's precision, testing the effectiveness of the search functionality, evaluation of developer interactions with bug reports, assessment of user experience and interface usability, measurement of application performance under varying loads, and checking compatibility across different web browsers and operating systems.

**Target Users:**

The target users for Bug Squasher include software development teams and individuals involved in bug reporting and management within a software development environment. These users may vary in technical expertise but share a common interest in effectively identifying, reporting, and addressing software bugs.

**Document Purpose:**

The purpose of this document is to outline the test plan for Bug Squasher, a gamified bug reporting tool. It serves as a guide for the testing process, detailing the objectives, scope, resources, schedule, and methodologies to be employed during testing. The document aims to ensure thorough testing coverage, adherence to quality standards, and the successful verification of Bug Squasher's functionality, usability, performance, and compatibility.

**References:**

1. Project Description Document

2. Functional Specification Document

3. User Interface Mockups

4. Testing Standards and Guidelines

**Test Scope:**

The test scope defines the boundaries of testing, outlining which features, functionalities, and aspects of the application will be included in the testing process. In the context of Bug Squasher, the test scope covers various components such as bug submission, severity levels, point and badge system, leaderboard, bug management, and search functionality. It ensures that all essential elements of the application are thoroughly tested to verify their correctness, reliability, and performance. The test scope helps in setting clear expectations for what will be tested and what will not, thus guiding the testing efforts effectively.

**Test Approach:**

The test approach outlines the overall strategy or methodology that will be employed during the testing process. It encompasses the techniques, tools, resources, and timelines to be utilized for conducting tests. In the case of Bug Squasher, the test approach may involve a combination of manual and automated testing techniques. It may also include strategies for prioritizing tests based on risk, ensuring comprehensive coverage, and maximizing efficiency. The test approach guides the testers in executing test cases, identifying defects, and evaluating the overall quality of the application. It ensures that testing activities are well-organized, systematic, and aligned with the project goals and objectives.

**1. Test Environment**

The test environment is a crucial component to ensure that the application is tested in conditions that closely mimic the production environment. This helps in identifying potential issues and bugs that might occur when the application is live. The following elements make up the test environment for the Bug Squasher project:

**Testing Tools**

**Selenium:**  Used for automated functional testing of the web application. Selenium scripts will be used to simulate user interactions with the application, ensuring that all functionalities are working as expected.

**JMeter:** Utilized for performance and load testing. JMeter will simulate multiple users accessing the application simultaneously to test the application's performance under load.

**Test Environment Setup**

**Staging Server:** A replica of the production server where the application will be deployed for testing. This server will have the same configurations, database, and network setup as the production environment to ensure accurate test results.

**Browser Compatibility:** Testing will be conducted across various web browsers, including Chrome, Firefox, Safari, and Edge, to ensure the application works seamlessly across different platforms.

**Operating Systems:** The application will be tested on multiple operating systems, including Windows, macOS, and Linux, to verify compatibility and performance across different environments.

**2. Roles and Responsibilities**

Effective testing requires a coordinated effort from the entire team. Clear roles and responsibilities ensure that each team member knows their tasks and contributes effectively to the project's success. Here are the key roles and their responsibilities for the Bug Squasher project:

**Test Lead**

**Planning:** Develop the overall test strategy and plan, ensuring all aspects of the application are covered.

**Task Assignment:**  Allocate tasks to testers, ensuring a balanced workload and that all areas are tested comprehensively.

**Progress Tracking:** Monitor the progress of testing activities, ensuring timelines are met and any issues are addressed promptly.

Reporting: Compile test results and generate summary reports to provide insights into the testing outcomes and application quality.

**Testers**

**Test Execution:** Execute the test cases as per the test plan, following the defined steps and documenting results.

**Defect Logging:** Identify and log defects in the bug tracking tool (Jira) with detailed information, including steps to reproduce, severity level, and screenshots if applicable.

**Verification:** Re-test fixed defects to ensure they are resolved and do not impact other parts of the application.

**Documentation:** Maintain detailed records of test cases executed, results, and any deviations or issues encountered.

**Developers**

**Bug Fixing:** Address and fix bugs reported by the testers, ensuring the application functions as expected.

**Code Reviews:** Conduct code reviews to identify potential issues and improve code quality before it reaches the testing phase.

**Collaboration:** Work closely with testers to understand reported defects, reproduce issues, and provide necessary fixes.

**Feature Implementation:** Develop and implement new features as per the project requirements, ensuring they are testable and meet the defined specifications.

**Project Manager (Optional):**

**Oversight:** Provide overall project oversight, ensuring alignment with project goals and timelines.

**Resource Allocation:** Ensure adequate resources (time, tools, personnel) are available for testing activities.

**Stakeholder Communication:** Maintain communication with stakeholders, providing updates on testing progress and application quality.

**1. Schedule**

The test schedule outlines the timeline for various testing phases:

| Testing Phase | Duration |

|----------------------------|----------|

| Functionality Testing | 10 days |

| Usability Testing | 5 days |

| Performance Testing | 5 days |

| Compatibility Testing | 3 days |

Each phase focuses on specific aspects of the application to ensure comprehensive testing and timely completion.

**2. Risk Management**

Risk management identifies potential risks and mitigation strategies:

Incomplete or Ambiguous Bug Reports

Mitigation: Provide a detailed bug report template and guidelines to ensure clarity and completeness.

**Server Downtime During Testing**

Mitigation: Schedule testing during off-peak hours and ensure backup servers are available to minimize disruptions.

3. Reporting

Reporting involves documenting and summarizing the test results:

**Test Summary Report:** A comprehensive document that includes:

* Number of test cases executed
* Number of defects found and fixed
* Performance and usability testing results
* Compatibility testing results

This report provides insights into the overall quality and readiness of the application.

**4. Approval**

Approval is the formal sign-off from stakeholders indicating that the application has met the required standards:

**Criteria for Approval:**

* All critical and major bugs must be resolved.
* No high-severity bugs should be present.
* All functionalities should work as expected without issues.
* The application should perform well under the expected load.

Stakeholders review the test summary report and approve the application for release if it meets the criteria

**5. Appendix**

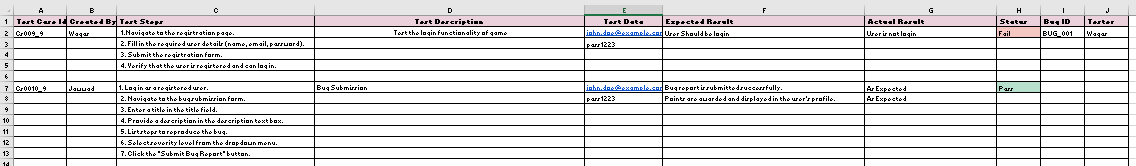
The appendix contains supplementary information:

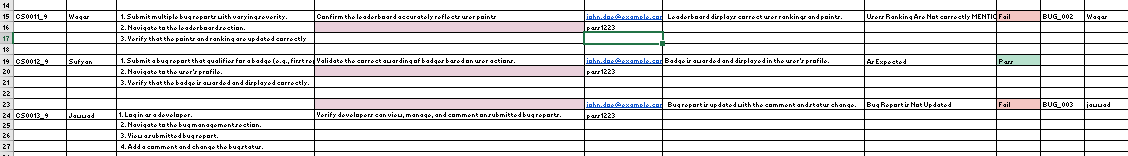
References: Project description documents, functional specifications, user interface mockups, and testing standards.

Glossary: Definitions of key terms and acronyms used in the test plan.

Test Data Samples: Examples of test data used during testing.

The appendix provides additional context and resources that support the main test plan.





**6. Test Cases**

Test cases are detailed steps used to verify specific functionalities:

Structure of Test Cases:

Test Case ID: Unique identifier for each test case.

-Description: Brief explanation of what the test case verifies.

Steps: Detailed steps to execute the test.

Expected Result: The expected outcome if the test passes.

Status: Indicates whether the test has passed or failed.

**1. Bug Submission Form Test Cases**

**Test Case ID:** TC-01  
**Description:** This test case verifies that a user can successfully submit a bug report with all the required fields completed.  
**Steps:**

1. Open the Bug Submission Form.
2. Fill in the Title field.
3. Fill in the Description field.
4. (Optional) Upload a Screenshot.
5. List the Steps to Reproduce the bug.
6. Select a Severity Level.
7. Click the Submit button.  
   **Expected Result:** The bug report is submitted successfully, and a confirmation message is displayed.

**Test Case ID:** TC-02  
**Description:** This test case verifies that a user can submit a bug report without including an optional screenshot.  
**Steps:**

1. Open the Bug Submission Form.
2. Fill in the Title field.
3. Fill in the Description field.
4. List the Steps to Reproduce the bug.
5. Select a Severity Level.
6. Click the Submit button.  
   **Expected Result:** The bug report is submitted successfully, and a confirmation message is displayed.

**2. Severity Levels Test Cases**

**Test Case ID:** TC-03  
**Description:** This test case verifies that a bug report can be submitted with a Critical severity level.  
**Steps:**

1. Open the Bug Submission Form.
2. Fill in all required fields (Title, Description, Steps to Reproduce).
3. Select "Critical" as the Severity Level.
4. Click the Submit button.  
   **Expected Result:** The bug report is submitted with a Critical severity level.

**Test Case ID:** TC-04  
**Description:** This test case verifies that a bug report can be submitted with a Major severity level.  
**Steps:**

1. Open the Bug Submission Form.
2. Fill in all required fields (Title, Description, Steps to Reproduce).
3. Select "Major" as the Severity Level.
4. Click the Submit button.  
   **Expected Result:** The bug report is submitted with a Major severity level.

**3. Point System Test Cases**

**Test Case ID:** TC-05  
**Description:** This test case verifies that points are awarded for submitting a detailed bug report with a screenshot.  
**Steps:**

1. Open the Bug Submission Form.
2. Fill in the Title field.
3. Fill in a detailed Description.
4. Upload a Screenshot.
5. List the Steps to Reproduce the bug.
6. Select a Severity Level.
7. Click the Submit button.  
   **Expected Result:** Points are awarded for submitting a detailed description and including a screenshot.

**Test Case ID:** TC-06  
**Description:** This test case verifies that points are awarded for submitting a bug report with a basic description.  
**Steps:**

1. Open the Bug Submission Form.
2. Fill in the Title field.
3. Fill in a basic Description.
4. List the Steps to Reproduce the bug.
5. Select a Severity Level.
6. Click the Submit button.  
   **Expected Result:** Points are awarded for the bug report submission.