Software testing standard

**ⅠDefect report**

1. **Defect Summary**:
   * A concise title or summary describing the issue.
2. **Defect Description**:
   * Detailed description of the problem, including observed behavior, expected behavior, and the environment in which the issue occurred (e.g., operating system, browser version).
   * Severity, impact, and potential risks associated with the issue.
3. **Steps to Reproduce**:
   * Clear instructions on how to reproduce the issue, allowing others to replicate it.
4. **Expected Results**:
   * Description of the expected behavior or outcome.
5. **Actual Results**:
   * Description of the actual behavior or outcome observed, compared to the expected results.
6. **Environment Details**:
   * Information about the environment in which the issue occurred, such as operating system, browser version, device model, etc.
7. **Additional Information**:
   * Any supplementary information like log files, screenshots, videos, etc., that may aid in identifying and resolving the issue.
8. **Severity**:
   * Assessment of the severity of the issue, typically using standard severity levels (e.g., critical, major, minor).
9. **Priority**:
   * Determination of the priority of the issue, typically using standard priority levels (e.g., high, medium, low).
10. **Reporter Information**:
    * Details of the person reporting the issue, including name, contact information, and date/time of the report.
11. **Assigned To**:
    * Assignment of the responsible individual or team for resolving the issue.
12. **Status**:
    * Tracking the current status of the issue, such as new, assigned, in progress, resolved, etc.
13. **Resolution**:
    * Description of the solution to the issue, including steps taken to fix it and the version in which it was resolved.
14. **Comments**:
    * Any additional comments or discussions related to the issue.

**Ⅱperformance test**

1. **Introduction**:
   * Brief overview of the application or system being tested.
   * Explanation of the purpose and scope of the performance testing.
   * Identification of the testing environment (hardware, software, network configurations).
2. **Objectives**:
   * Clear listing of the objectives of the performance testing, such as response times, throughput, concurrency levels, etc.
3. **Test Approach**:
   * Description of the testing methodology and strategy, including tools used, test environment setup, and design of test cases.
   * Explanation of why specific performance testing methods and metrics were chosen.
4. **Test Environment**:
   * Detailed description of the hardware, software, and network environment used for testing.
5. **Test Execution**:
   * Description of the actual tests performed, including test dates, duration, test steps, and any issues or anomalies encountered.
   * Record of all test parameters and configurations.
6. **Performance Metrics**:
   * Summarization of performance test results, including response times, throughput, resource utilization, etc.
   * Provision of charts or graphs to visualize the data for better understanding.
7. **Results Analysis**:
   * Analysis of performance testing data, explaining whether the test results meet the expected objectives.
   * Identification of system performance bottlenecks and potential issues.
   * Comparison and evaluation of results, possibly including comparisons with previous test results or competitor products.
8. **Issues and Recommendations**:
   * Listing of any issues or potential risks discovered during testing.
   * Provision of improvement recommendations to address performance issues or optimize system performance.
9. **Conclusion**:
   * Summary of the testing results and analysis.
   * Restatement of the testing objectives and discussion of whether those objectives were achieved.
   * Proposal of future work or testing suggestions that may be needed.
10. **Appendices**:

* Inclusion of additional supporting information such as test scripts, configuration files, detailed raw data, etc.

|  |  |
| --- | --- |
| Test start date |  |
| Test end date |  |
| Number of submitted defects |  |
| Number of Deferred/Unresolved Defects |  |
| Rejected defect number |  |
| Total test scenario |  |
| Number of execution scenarios |  |
| Scene execution rate |  |
| Defect rate (%) |  |
| Input number |  |
| Test productivity |  |
| System online suggestion |  |

**Ⅲstatic analysis**

1. **Introduction**:
   * Overview of the static analysis process.
   * Purpose of static analysis.
   * Scope of the analysis (e.g., specific modules, entire codebase).
2. **Static Analysis Tools**:
   * Description of the tools used for static analysis (e.g., linting tools, code quality analyzers, security scanners).
   * Explanation of why these tools were chosen.
3. **Analysis Results**:
   * Summary of the analysis findings.
   * Key metrics and statistics obtained from the analysis (e.g., number of issues found, severity levels).
   * Categorization of issues (e.g., code style violations, security vulnerabilities, performance bottlenecks).
4. **Detailed Findings**:
   * Detailed breakdown of each type of issue identified, including:
     + Description of the issue.
     + Location in the codebase (file, line numbers).
     + Severity level.
     + Potential impact.
     + Recommendations for resolution.
5. **Code Quality Metrics**:
   * Overview of code quality metrics obtained from the analysis.
   * Explanation of each metric (e.g., cyclomatic complexity, code duplication).
   * Comparison of metrics against industry standards or best practices.
6. **Security Analysis**:
   * Summary of security-related findings from the static analysis.
   * Types of vulnerabilities identified (e.g., SQL injection, cross-site scripting).
   * Severity levels of vulnerabilities.
   * Recommendations for mitigating security risks.
7. **Performance Analysis**:
   * Overview of performance-related findings from the static analysis.
   * Identification of performance bottlenecks or inefficient code patterns.
   * Recommendations for improving performance.
8. **Integration and Build Process**:
   * Explanation of how static analysis is integrated into the development workflow.
   * Description of any automated processes for running static analysis during builds or code reviews.
9. **Conclusion**:
   * Summary of the overall findings and observations from the static analysis.
   * Assessment of the code quality, security posture, and performance of the software.
   * Recommendations for further actions (e.g., code refactoring, security patches, performance optimization).
10. **References**:
    * Any external references or sources consulted during the static analysis process

**Ⅳcoverage**

1. **Introduction**:
   * Overview of the purpose and scope of the coverage document.
   * Explanation of the concept of test coverage and its importance for software quality and testing effectiveness.
2. **Coverage Metrics**:
   * Detailed explanation of the coverage metrics used, such as statement coverage, branch coverage, path coverage, etc.
   * Explanation of each metric, including its calculation method and significance.
3. **Scope of Testing**:
   * Determination of the scope of test coverage, whether it's for the entire codebase or specific modules, features, or requirements.
   * Description of the relevant information about the target being tested, such as code version, functional specifications, requirements documents, etc.
4. **Testing Strategy**:
   * Explanation of the testing coverage strategy and methods, including principles and techniques for test case design.
   * Discussion on how to ensure that test cases effectively cover code, features, or requirements.
5. **Coverage Analysis Results**:
   * Provision of the actual coverage analysis results.
   * Summarization of the data for each coverage metric, including coverage percentages and specific coverage details.
6. **Coverage Gaps**:
   * Description of any deficiencies or gaps identified in the coverage analysis.
   * Identification of areas with lower coverage rates and explanation of possible reasons and potential risks.
7. **Recommendations**:
   * Suggestions for improving test coverage, such as adding test cases, enhancing test design, optimizing test processes, etc.
8. **Conclusion**:
   * Summary of the main findings and results of the coverage document.
   * Emphasis on the importance of test coverage for software quality and stability.
   * Proposals for future improvement directions and continuous improvement strategies.
9. **Appendices**:
   * Additional supporting information, such as test case lists, screenshots of coverage reports, specific coverage analysis data, etc

**Ⅴsystem test**

1. **Introduction**:
   * Overview of the system testing process.
   * Purpose and objectives of system testing.
   * Scope of testing and the systems under test.
2. **Test Plan**:
   * Detailed plan outlining the approach, resources, schedule, and responsibilities for system testing.
   * Test objectives, test deliverables, and acceptance criteria.
3. **Test Cases**:
   * Description of test cases designed for system testing.
   * Test case ID, description, test steps, expected results, actual results, and pass/fail status.
4. **Test Execution**:
   * Description of the actual test execution process.
   * Test environment setup details.
   * Execution of test cases, including any deviations or issues encountered.
5. **Test Results**:
   * Summary of test results, including pass/fail status for each test case.
   * Metrics such as test coverage, defect density, etc.
   * Detailed analysis of any failed test cases, including root cause analysis.
6. **Defects**:
   * Comprehensive list of defects found during system testing.
   * Defect ID, description, severity, priority, status, and resolution details.
   * Classification of defects based on severity levels.
7. **Traceability Matrix**:
   * Mapping between requirements, test cases, and test results.
   * Ensures that all requirements have been covered by test cases and that all test cases have been executed.
8. **Risk Assessment**:
   * Assessment of risks identified during system testing.
   * Mitigation strategies for high-risk areas.
9. **Conclusions and Recommendations**:
   * Summary of overall system testing results.
   * Lessons learned and recommendations for future testing efforts.
   * Any outstanding issues or areas requiring further investigation.
10. **Appendices**:
    * Additional supporting documentation such as test data, test scripts, test logs, etc.