**What is Software Testing?**

Software testing is an important process in the software development lifecycle.

It involves verifying and validating that a software application is free of bugs,

meets the technical requirements set by its design and development, and satisfies

user requirements efficiently and effectively.

This process ensures that the application can handle all exceptional and

boundary cases, providing a robust and reliable user experience.

By systematically identifying and fixing issues, software testing helps deliver

high-quality software that performs as expected in various scenarios.

**Basic Terminologies of Software Testing**

[Home](https://www.appsierra.com/) >> [Blogs](https://www.appsierra.com/blogs) >> Basic Terminologies of Software Testing

There are some kinds of terminologies for every industry. Below is a comprehensive list of over 150 software testing terminologies , including [manual QA testing solutions](https://www.appsierra.com/blog/manual-testing-services), that are used frequently in the working of this industry. This list also defines software testing. These are the basic testing concepts. Hope you learn a lot!

**Software Testing Terminologies**

Here are some software testing terminologies that are widely and popularly used in the [software testing](https://www.appsierra.com/blog/software-testing-services) and Information Technology industries:

**1. Acceptance Testing**

Formal testing of user needs, requirements, and business processes is conducted to determine whether or not a system meets the acceptance criteria and to allow the user, customers, or other authorized entity to determine whether or not to accept the system.

**2. Alpha Testing**

Simulated or actual operational testing by potential users/customers or an independent testing team on the developers’ website, but outside the development organization. Thus, manual testing means alpha testing. Alpha testing is often employed as a form of the internal acceptance test.

**3. Attack**

Directed and focused attempt on quality assessment, especially reliability, of a test object, trying to force specific failures.

**4. Beta Testing**

Operational testing by users / potential customers and/or existing on an external site that is not involved with developers. Its purpose is to determine whether a component or system meets the needs of the user/client and fits within the business processes. [Beta testing](https://www.appsierra.com/blog/what-is-beta-testing) is often employed as a form of external acceptance testing to get market feedback.

**5. Black Box Testing**

Functional or non-functional tests, without reference to the internal structure of the component or system.

**6. Boundary Value Analysis**

A black box test design technique in which test cases are designed based on threshold values.

**7. Branch Coverage**

The percentage of branches that were evaluated by a set of tests. One hundred percent affiliate coverage implies 100% decision coverage and 100% declaration coverage.

**8. Code Coverage**

An analysis method that determines which parts of the software were run (covered) by the test suite and which parts were not executed.

**9. Compiler**

A software tool that translates programs expressed in a machine language.

**10. Complexity**

The degree to which a component or system has an internal design and/or structure that is difficult to understand, maintain, and verify.

**11. Component Integration Testing**

Tests were performed to identify defects in interfaces and interactions between integrated components.

**12. Component Testing**

Testing individual software components.

**13. Configuration Control**

A configuration management element that consists of evaluating, coordinating, approving, or disapproving, and implementing changes to configuration items after the formal establishment of your configuration ID.

**14. Configuration Item**

An aggregation of hardware, software, or both, which is intended for configuration management and treated as a single entity in the configuration management process.

**15. Configuration Management**

A discipline that applies direction, technical, and administrative surveillance to identify and document the functional and physical characteristics of a configuration item, track changes in these characteristics, record and report change processing and implementation status, and verify compliance with the specified requirements.

**16. Control Flow**

An abstract representation of all possible sequences of events and paths in execution through a component or system.

**17. Coverage**

The grade is expressed as a percentage, to which a specified coverage item was run by a test suite.

**18. Coverage Tool**

A tool that provides objective measurements of which structural elements were executed by the test suite.

**19. Cyclomatic Complexity**

The number of independent paths through a program.

**20. Data-Driven Testing**

A scripting technique that stores test input and expected results in a table or worksheet so that a single control script can run all tests in the table. Data tests are often used to support the application of test execution tools such as capture/playback tools.

**21. Data Flow**

An abstract representation of the sequence and possible changes in the state of data objects, where the state of an object is any creation, use, or destruction.

**22. Debugging**

The process of finding, analyzing, and removing the causes of software failures.

**23. Debugging Tool**

A tool used by programmers to reproduce failures, investigate the state of programs, and find the corresponding defect. Debuggers allow programmers to run programs step by step, stop a program in any program instruction, and define and examine program variables.

**24. Decision Coverage**

The percentage of decision results that were exercised by a set of tests. One hundred percent decision coverage implies both 100% affiliate coverage and 100% declaration coverage.

**25. Decision Table Testing**

A black box testing technique in which test cases are designed to perform the combinations of inputs and/or stimuli (causes) shown in a decision table.

**26. Defect**

A failure in a component or system can cause the component or system to fail when performing its required function. An incorrect declaration or data definition. A defect, if found during execution, can cause a component or system failure.

**27. Defect Density**

The number of defects identified in a component or system is divided by the size of the component or system (expressed in terms of standard measurement, for example, lines of code, number of classes, or function points).

**28. Defect Management**

The process of recognition, investigation, action, and elimination of defects. It involves recording defects, classifying them, and identifying their impact. It is a very important software testing terminology.

**29. Driver**

A software component or test tool that replaces a component that takes care of the control and/or calls of a component or system.

**30. Dynamic Analysis Tool**

A tool that provides run-time information about the state of the software code. These tools are most commonly used to identify unassigned pointers, check pointer arithmetic, monitor memory allocation, usage, and distribution, and highlight memory leaks.

**31. Dynamic Testing**

Testing that involves running the system so that it is dynamically tested.

**32. Entry Criteria**

The set of generic and specific conditions allows a process to proceed with a defined task.

**33. Error Guessing**

Another important software testing terminology is error guessing. It is a test design technique where the tester experience is used to anticipate which defects may be present in the component or system under test as a result of errors made and design tests specifically to expose them.

**34. Exit Criteria**

The set of generic and specific conditions agreed with stakeholders to allow a process to be officially completed. The purpose of the exit criteria is to prevent a task from being considered completed when there are still pending parts of the task that have not been completed. The output criteria are used by tests to report against and plan when to stop the test.

**35. Exhaustive Testing**

A test approach in which the test suite comprises all combinations of input values and preconditions.

**36. Exploratory Testing**

Test where the tester actively controls the design of tests when these tests are performed and uses the information obtained during testing to design new and better tests.

**37. Failure**

Actual deviation of the component or system from its expected delivery, service, or result. The inability of a system or component to perform a required function within specified limits.

**38. Failure Rate**

The proportion of the number of failures in a given category for a given unit of measure, e. Failures per unit of time, failures by the number of transactions, and failures by the number of computer runs. This software testing terminology is very crucial.

**39. Formal Review**

A review characterized by documented procedures and requirements, e.g. inspection.

**40. Functional Requirement**

A requirement that specifies a function that a component or system must perform.

**41. Functional Testing**

Testing is based on an analysis of the specification of the functionality of a component or system.

**42. Horizontal Traceability**

Tracking requirements for a test level through the test documentation layers (for example, test plan, test project specification, test case specification, and test procedure specification).

**43. Impact analysis**

The evaluation of the change to the development documentation layers, test documentation, and components to implement a particular change to the specified requirements.

**44. Incident**

Any event that occurs during the test that requires investigation.

**45. Incident Report**

A document that reports any event that occurs during testing, which requires investigation.

**46. Incident Management Tool**

The tool facilitates the recording and status monitoring of incidents encountered during testing. They often have workflow-oriented capabilities to track and track incident allocation, remediation, and retesting, and provide reporting facilities.

**47. Incremental Development Model**

A development lifecycle in which a project is divided into a series of increments, each of which provides a piece of functionality in the overall requirements of the project. Requirements are prioritized and delivered in order of priority in the appropriate increment.

In some (but not all) versions of this lifecycle model, each subproject follows a “mini-model V” with its design, coding, and testing phases.

**48. Independence of Testing**

Separation of responsibilities, which encourages the performance of objective tests.

**49. Informal review**

A review not based on a formal procedure (documented).

**50. Inspection**

A type of review that relies on visual examination of documents to detect defects. Another important software testing terminology.

**51. Intake Test**

A special example of a smoke test is to decide whether the component or system is ready for detailed and additional testing.

**52. Integration**

The process of combining components or systems into larger assemblies.

**53. Integration Tests**

Tests are performed to expose defects in interfaces and integrations between components or integrated systems.

**54. Interoperability Testing**

The testing process for determining the interoperability of a software product

**55. ISTQB**

International Software Testing Qualifications Board, a non-profit association that develops international certification for software testers. An important software testing terminology.

**56. Keyword-Driven Testing**

A scripting technique that uses data files to contain not only test data and expected results but also keywords related to the application being tested. Keywords are interpreted by special support scripts that are called by the control script for testing. See also data tests.

**57. Maintainability Testing**

The testing process for determining the maintenance of a software product.

**58. Metric**

A measurement scale and method used for measurement.

**59. Moderator**

The leader and principal are responsible for an inspection or other review process.

**60. Modeling Tool**

The tool that supports the validation of software or system models.

**61. N-Switch Coverage**

The percentage of N+1 transition sequences that were exercised by a set of tests.

**62. N-Switch Testing**

A form of state transition testing in which test cases are designed to perform all valid sequences of N+1 transitions.

**63. Non-Functional Requirement**

Requirement that does not refer to functionality, but attributes of it, such as reliability, efficiency, ease of use, maintenance, and portability.

**64. Off-the-shelf Software**

A software product that is developed for the general market, that is, for a large number of customers and that is delivered to many customers in an identical format.

**65. Performance Testing**

The testing process for determining the performance of a software product.

**66. Performance Testing Tool**

A tool to support performance testing that typically has two main facilities: load generation and test transaction measurement. A load generation can simulate multiple users or large volumes of input data.

During execution, response time measurements are taken from selected transactions and are recorded. Performance testing tools typically provide reports based on test records and load graphs against response times.

**67. Portability Testing**

The testing process for determining the portability of a software product.

**68. Probe Effect**

The effect on the component or system when it is being measured, e.g. by a performance test tool or monitor.

**69. Product Risk**

Risk is directly related to the test object.

**70. Project Risk**

A risk related to project management and control. E.g. lack of staff, strict deadlines, change of requirements, etc.

**71. Project Test Plan**

A test plan that typically meets multiple test levels.

**72. Quality**

The degree to which a component, system, or process meets the requirements and/or needs and expectations of users/customers.

**73. RAD**

Rapid Application Development, a software development model.

**74. Regression Testing**

Testing a previously tested after-modification program to ensure that defects have not been introduced or discovered in unchanged areas of the software as a result of changes made. It runs when the software or its environment changes.

**75. Reliability Testing**

The testing process for determining the reliability of a software product.

**76. Requirement**

A condition or capacity required by the user to resolve a problem or achieve a goal that must be achieved or possessed by a component system or system to satisfy a contract, standard, specification, or another formally imposed document.

**77. Requirement Management Tool**

A tool that supports the recording of requirements, requirements attributes (for example, priority, responsible knowledge), and annotation and facilitates traceability through requirements layers and requirements change management. Some requirements management tools also provide facilities for static analysis, such as consistency checking and violations of predefined requirements rules.

**78. Re-Testing**

Test that runs test cases that failed the last time they were run to verify the success of corrective actions.

**79. Review**

An evaluation of a product or project status to verify discrepancies in planned results and recommend improvements. Examples include management review, informal review, technical review, inspection, and walk-through.

**80. Review Tool**

A tool that supports the review process. Typical features include review planning and tracking, communication support, collaborative reviews, and a repository for collecting and reporting metrics. It helps to understand the definition of software testing.

**81. Reviewer**

The person involved in the review who identifies and describes anomalies in the product or project under review. Reviewers can be chosen to represent different views and roles in the review process.

**82. Risk**

A factor that could result in future negative consequences is generally expressed as impact and probability.

**83. Risk-Based Testing**

An approach to testing to reduce the level of product risk and inform stakeholders about its status, starting in the early stages of a project. It involves identifying the risks of the product and its use in guiding the testing process.

**84. Robustness Testing**

Testing to determine the robustness of the software product.

**85. SBTM**

Session-based test management, ad hoc, and exploratory test management techniques, based on fixed-duration sessions (from 30 to 120 minutes) during which testers exploit part of the application.

**86. Scribe**

The person who has to record each defect mentioned and any suggestions for improvement during a review meeting, on a [registration form](https://www.jotform.com/form-templates/category/registration). The scribe shall ensure that the registration form is legible and understandable.

**87. Scripting Language**

A programming language in which executable test scripts are written, and used by a test execution tool (for example, a capture/replay tool).

**88. Security Testing**

Testing to determine software product security.

**89. Site Acceptance Testing**

User/customer acceptance testing on your site to determine whether a component or system meets user/customer needs and fits within business processes, typically including hardware and software.

**90. SLA**

A service level agreement is a service agreement between a vendor and its customer, defining the level of service that a customer can expect from the provider.

**91. Smoke Test**

A subset of all defined/planned test cases that cover the core functionality of a component or system, to verify that the most crucial functions of a program work, but do not bother with more precise details. The daily test of construction and smoke is among the best practices in the industry.

**92. State Transition**

A transition between two states of a component or system.

**93. State Transition Testing**

A black box testing technique in which test cases are designed to perform valid and invalid state transitions.

**94. Statement Coverage**

The percentage of executable claims that have been exercised by a set of tests.

**95. Static Analysis**

Analysis of software artifacts, e.g. requirements or code, performed without the execution of these software artifacts.

**96. Static Testing**

The tool that performs static code analysis. The tool checks source code for certain properties, such as compliance with coding standards, quality metrics, or data flow anomalies.

**97. Static Testing**

Testing a component or system at the specification or implementation level without running that software, e.g. static code analysis.

**98. Stress Testing**

A type of performance test conducted to evaluate a system or component within the limits of its expected or specified workloads or with reduced availability of resources, such as memory access or servers. See also performance tests, and load tests.

**99. Stress Testing Tool**

A tool that supports stress testing.

**100. Structural Testing**

See white-box tests.

**101. Stub**

A skeletal or special-purpose implementation of a software component, used to develop or test a component that calls or is otherwise dependent on it. It replaces a component called.

**102. System Integration Testing**

Testing the integration of systems and packages; Testing interfaces for external organizations (e.g. electronic data exchange, internet).

**103. System Testing**

The process of testing an integrated system to verify that it meets the specified requirements.

**104. Technical Review**

A peer group discussion activity that focuses on reaching a consensus on the technical approach to be taken. A technical review is also known as a peer review.

**105. Test**

A set of one or more test cases.

**106. Test Approach**

The implementation of the test strategy for a specific project. Typically, includes the decisions made below based on the project goal (test) and the risk assessment performed, starting points on the test process, the test design techniques to be applied, the exit criteria, and the types of tests to be performed.

**107. Test Basis**

All documents from which the requirements of a component or system can be inferred. The documentation on which test cases are based. If a document can only be amended utilizing a formal amending procedure, the test basis is called the frozen test basis.

**108. Test Case**

A set of input values run preconditions, expected results, and post-execution conditions, developed for a particular test objective or condition, such as exercising a specific program path or verifying compliance with a specific requirement.

**109. Test Case Specification**

A document specifying a set of test cases (objective, inputs, test actions, expected results, and execution conditions) for a test item.

**110. Test Comparator**

A test tool for comparing automated tests.

**111. Test Condition**

An item or event of a component or system that could be verified by one or more test cases, e.g. a function, transaction, quality attribute, or structural element.

**112. Test Control**

A test management task that handles the development and application of a set of corrective actions to get a test project on track when monitoring shows a deviation from the plan.

**113. Test Coverage**

See the coverage.

**114. Test Data**

Data that exists (for example, in a database) before a test is run, and that affects or is affected by the component or system being tested.

**115. Test Data Preparation Tool**

A type of test tool that allows data to be selected from existing databases or created, generated, manipulated, and edited for use in tests.

**116. Test Design**

The process of transforming overall test objectives into tangible test conditions and test cases.

**117. Test Design Specification**

A document specifying the test conditions (coverage items) for a test item, such as a detailed test approach and identification of associated high-level test cases.

**118. Test Design Technique**

A method used to derive or select test cases.

**119. Test Design Tool**

A tool that supports test project activity by generating test inputs from a specification that can be maintained in a CASE tool store, e.g. requirements management tools, or from specified test conditions maintained in the tool itself.

**120. Test-Driven Development**

Agile development method, where tests are designed and automated before code (from requirements or specifications), then the minimum amount of code is written to successfully pass the test. This method is iterative and ensures that the code continues to meet the requirements through the test run.

**121. Test Environment**

An environment containing hardware, instrumentation, simulators, software tools, and other supporting elements needed to perform a test.

**122. Test Execution**

The process of running a test by the component or system under test, producing actual results.

**123. Test execution Schedule**

A schema for running test procedures. Test procedures are included in the test execution schedule in its context and in the order in which it should be run.

**124. Test Execution Tool**

A type of test tool that can run other software using an automated test script, e.g. Capture/Playback.

**125. Test Harness**

A test environment composed of stubs and drivers is required to perform a test.

**126. Test Leader**

Consult the Test manager.

**127. Test Level**

A group of test activities that are organized and managed together. A test level is linked to the responsibilities of a project.

**128. Test Log**

A chronological record of relevant details about running tests.

**129. Test Management**

The planning, estimation, monitoring, and control of tests and activities, are usually performed by a test manager.

**130. Test Manager**

The person is responsible for testing and evaluating a test object. The individual, who manages, controls, manages, plans, and regulates the evaluation of a test object.

**131. Test Monitoring**

A test management task that handles activities related to periodic verification of the status of a test project. The reports are prepared that compare the results with what was expected.

**132. Test Objective**

A reason or purpose for designing and running a test.

**133. Test Oracle**

A source to determine the expected results to compare with the actual result of the software under test. An oracle can be the existing system (for a benchmark), a user manual, or the specialized knowledge of an individual, but it should not be coded.

**134. Test Plan**

A document describing the scope, approach, features, and schedule of the intended test activities. Among others, it identifies the test items, the resources to be tested, the test tasks, who will perform each task, the degree of independence of the tester, the test environment, the test techniques, and the test measurement techniques to be used. It is a record of the test planning process.

**135. Test Policy**

A high-level document that describes the principles, approach, and key objectives of the organization concerning testing.

**136. Test Procedure Specification**

A document specifying a sequence of actions for running a test; Also known as a test script or manual test script.

**137. Test Script**

Commonly used to refer to a test procedure specification, especially an automated one.

**138. Test Strategy**

A high-level document that defines the test levels to be run and the test within those levels for a program (one or more projects). A test strategy helps to know the definition of software testing.

**139. Test Suite**

A set of multiple test cases for one component or system under test, where the postcondition condition of one test is often used as the precondition for the next.

**140. Test Summary Report**

A document that summarizes the test activities and results. It also contains an evaluation of the corresponding test items against the output criteria.

**141. Tester**

A technically qualified professional who is involved in testing a component or system.

**142. Testware**

Artifacts produced during the testing process are required to plan, design, and run tests, such as documentation, scripts, inputs, expected results, installation and cleanup procedures, files, databases, environment, and any additional software or utilities used in testing.

**143. Thread Testing**

A component integration test version where the progressive integration of components follows the implementation of subsets of requirements, as opposed to integrating components by levels of a hierarchy.

**144. Traceability**

The ability to identify related items in documentation and software, such as requirements with associated tests.

**145. Usability Testing**

Test to determine the extent to which the software product is understood, easy to learn, easy to operate, and attractive to users under specified conditions. By knowing the definition of software testing, you can implement usability testing.

**146. Use Case Testing**

A black box test design technique in which test cases are designed to run user scenarios.

**147. User Acceptance Testing**

See acceptance testing test.

**148. V-Model**

A framework to describe the life cycle activities of software development, from requirements specification to maintenance. Model V illustrates how to test activities that can be integrated into each phase of the software development lifecycle.

**149. Validation**

Confirmation by testing and by providing objective evidence that the requirements for a specific use or application have been met.

**150. Verification**

Confirmation by testing and by providing objective evidence that the specified requirements have been met.

**151. Vertical Traceability**

The tracing of requirements through the development of documentation layers for components.

**152. Walk-Through**

A step-by-step presentation by the author of a document to gather information and establish a common understanding of its content.

**153. White-Box Testing**

Testing is based on an analysis of the internal structure of the component or system.

**Conclusion**

So, aforementioned is a list of software testing terminologies. Be sure to mention all terms we may have missed in the comments below!