**Test case**-A test case is a singular set of actions or instructions for a tester to perform that validates a specific aspect of a product or application functionality. If the test fails, the result might be a software defect that the organization can triage.

A tester or QA professional typically writes test cases, which are run after the completion of a feature or the group of features that make up the release. Test cases also confirm whether the product meets its software requirements.

**Types of test cases**

To validate and verify system functionality, the organization must take a multi-faceted approach that evaluates the product’s front and back ends. There are different ways to categorize the various types of test cases. One way to start is with these two categories: formal and informal.

**Formal test cases**. With these types of test cases, the tester writes a test in which the inputs are all known and detailed, such as the preconditions and test data. Formal tests have predefined input, which means they provide an expected output, which the test attempts to validate.

**Informal test cases**. Conversely, informal test cases do not have known inputs or outputs. Testers execute these types of test cases to discover and record the outcomes, which can reveal interesting findings about digital quality.

Most types of test cases are formal — planned in advance according to software requirements. Let’s explore some more test case types and examples:

* functionality
* UI
* integration
* performance
* usability
* database
* user acceptance
* exploratory

**Functionality test cases**. These tests determine whether the target functionality succeeds or fails to perform its function within the system. The QA team writes these types of test cases based on requirements and performs them when the dev team is finished with the function. Many different [types of functional tests](https://www.applause.com/blog/functional-testing-types-examples) can validate app functionality, including unit tests that check the smallest, isolated segments of functionality possible. Functional test cases should include:

* a description and/or name of the function under test
* preconditions
* steps for testing
* an expected result

Functionality test case example: Perform a successful login and validate that the user is logged in.

**UI test cases**. These tests confirm the user interface (what the end user interacts with) functions as expected. Typically, UI tests focus on an app or web page’s visual elements to confirm they function and perform according to requirements. UI tests often examine display elements such as menus, sub-menus, buttons, tables and columns to make sure they are readable and consistent.

UIs continue to evolve. For this reason, UI tests can also mean validating a voice or video interface. UI tests should also include accessibility concerns, such as whether a screen reader can identify a button on a page.

UI test case example: Navigate to the home page, validate that the hamburger menu displays correctly for desktop and mobile web.

**Integration test cases**. These types of test cases assess how the combined functionality works when merged into the application. While it is important to test individual units of software, it is equally important to make sure disparate systems can communicate with each other effectively. The tester must understand the application flows well to write effective integration tests.

API testing is one aspect of integration testing. Applications communicate with each other through APIs, especially as products become more interconnected in today’s mobile-centric world. API testing is a vital exercise to cover with integration test cases.

Integration test case example: Log in via a seller’s marketplace, validate that the marketplace then recognizes the user as logged in — in other words, the login and marketplace modules communicate with each other.

**Performance test cases**. Functional tests check whether the application works. Non-functional tests, such as performance testing, check how the application performs under different types of workloads. A performance test must be specific with each step and expected result documented, as well as input data clearly defined, so that the tester can accurately assess how the system performs in the given conditions.

There are a variety of performance testing types, including load, stress, spike and scalability testing. Each type of performance testing, and each individual test, reveals different information about how the system responds to varying user loads.

Performance test case example: Measure the largest number of users a system can handle before it crashes.

**Security test cases**. These tests identify vulnerabilities within a system or product. Another type of non-functional testing, security tests aim to find ways to better protect software assets, as well as identify how the system holds up against common types of attacks, and define the risk associated with the product.

Some security tests might include vulnerability scanning, configuration scanning and penetration testing, also called intrusive testing. Ultimately, the point of security testing is to yield actionable feedback that the organization can use to remediate vulnerabilities.

Security test case example: Validate that you cannot access company documents without a successful login.

**Usability test cases**. Rather than test the application functionality or performance, usability tests examine what prospective end users — not testers — think of a product. UX researchers prepare tests for participants outside the organization to gauge how easy or difficult the product is to use.

Organizations can [conduct usability testing](https://www.applause.com/blog/what-is-usability-testing-types-benefits) in a variety of ways, including moderated or unmoderated and remote or in-person. The goal is to take advantage of an end user’s perspective to identify points in the application that would cause them to stop using it. Usability tests can be formal or informal, depending on the goal and method of UX research.

Usability test case example: Task the participant with a money transfer between their checking and savings accounts, then gauge whether they can successfully complete the task and whether they experience any difficulty with the process.

**Database test cases**. Just because an app’s functionality, the user interface and APIs are all working doesn’t mean the data is being stored properly. Database tests validate whether the application data is stored in accordance with requirements and regulations. Like functionality tests, database tests can vary in scope, from validation of a small database object to a complex action involving multiple parts of the application.

Some criteria that database tests might evaluate include whether the data is stored consistently, whether unauthorized people can access it, and how it is stored locally on a device. Consistent and secure data should be a priority for every business, regardless of the industry’s compliance standards — database tests help achieve that.

Database test case example: Validate that new customer PII data is stored in an encrypted format.

**User acceptance test cases**. These types of test cases validate the product from the end user’s perspective. An end user or client conducts user acceptance tests in a testing environment to validate the end-to-end flow of the product.

User acceptance tests can come in handy when business requirements change during the course of development. Stakeholders do not always effectively communicate these changes to the dev team. Through UAT test cases, the organization can document entry and exit criteria that cover gaps in previous tests.

User acceptance test case example: Validate that a user can register for a new account and that they receive an email confirmation.

**Test scenario**-A **Test Scenario** is defined as any functionality that can be tested. It is also called *Test Condition* or *Test Possibility*. As a tester, you should put yourself in the end user’s shoes and figure out the real-world scenarios and use cases of the Application Under Test.

**Scenario Testing** in software testing is a method in which actual scenarios are used for testing the software application instead of test cases. The purpose of scenario testing is to test end to end scenarios for a specific complex problem of the software. Scenarios help in an easier way to test and evaluate end to end complicated problems.

**What is Test Basis?**

Test basis is defined as the source of information or the document that is needed to write test cases and also for test analysis. Test basis should be well defined and adequately structured so that one can easily identify test conditions from which test cases can be derived.

Typical Test Basis:

* Requirement document
* Test Plan
* Codes Repository
* Business Requirement

What is a Test Suite?

Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status. It can take any of the three states namely Active, Inprogress and completed.

A Test case can be added to multiple test suites and test plans. After creating a test plan, test suites are created which in turn can have any number of tests.

Test suites are created based on the cycle or based on the scope. It can contain any type of tests, viz - functional or Non-Functional.

Test Suite - Diagram:

