**Test Case**

Software testing is known as a process for [validating and verifying](https://www.geeksforgeeks.org/software-engineering-verification-and-validation/) the working of a software/application. It makes sure that the software is working without any errors, bugs, or any other issues and gives the expected output to the user. The software testing process doesn’t limit to finding faults in the present software but also finding measures to upgrade the software in various factors such as efficiency, usability, and accuracy. So, to test software the software testing provides a particular format called a **Test Case**.

This article focuses on discussing the following topics in the Test Case:

1. **What is a Test Case?**
2. **Test Case Vs Test Scenario.**
3. **When do we Write Test Cases?**
4. **Why Write Test Cases?**
5. **Test Case Template.**
6. **Best Practice for Writing Test Cases.**
7. **Test Case Management Tools.**
8. **Formal and Informal Test Case.**
9. **Types of Test Cases.**
10. **Example.**

**What is Test Case?**

A test case is a defined format for software testing required to check if a particular application/software is working or not. A test case consists of a certain set of conditions that need to be checked to test an application or software i.e. in more simple terms when conditions are checked it checks if the resultant output meets with the expected output or not. A test case consists of various parameters such as Id, condition, steps, input, expected result, result, status, and remarks.

**Parameters of a Test Case:**

* **Module Name:**Subject or title that defines the functionality of the test.
* **Test Case Id:**A unique identifier assigned to every single condition in a test case.
* **Tester Name:**The name of the person who would be carrying out the test.
* **Test scenario: The test** scenario provides a brief description to the tester, as in providing a small overview to know about what needs to be performed and the small features, and components of the test.
* **Test Case Description:** The condition required to be checked for a given software. for eg. Check if only numbers validation is working or not for an age input box.
* **Test Steps:** Steps to be performed for the checking of the condition.
* **Prerequisite:**The conditions required to be fulfilled before the start of the test process.
* **Test Priority:**As the name suggests gives the priority to the test cases as in which had to be performed first, or are more important and which could be performed later.
* **Test Data:** The inputs to be taken while checking for the conditions.
* **Test Expected Result:** The output which should be expected at the end of the test.
* **Test parameters:**Parameters assigned to a particular test case.
* **Actual Result:** The output that is displayed at the end.
* **Environment Information:**The environment in which the test is being performed, such as the operating system, security information, the software name, software version, etc.
* **Status:** The status of tests such as pass, fail, NA, etc.
* **Comments:** Remarks on the test regarding the test for the betterment of the software.

**Test Case vs Test Scenario**

Below are some of the points of difference between a test case and a test scenario:

| **Parameter** | **Test Case** | **Test Scenario** |
| --- | --- | --- |
| **Definition** | A test case is a defined format for software testing required to check if a particular application/software/module is working or not. Here we check for different conditions regarding the same. | The test Scenario provides a small description of what needs to be performed based on the use case. |
| **Level of detailing** | Test cases are more detailed with a number of parameters. | Test Scenario provides a small description, mostly one-line statements. |
| **Action Level** | Test cases are low-level actions. | Test scenarios are high-level actions. |
| **Derived from** | Test cases are mostly derived from test scenarios. | Test scenarios are derived from documents like BRS, SRS, etc. |
| **Objective** | It focuses on “What to test” and “How to test”. | It focuses more on ‘What to test”. |
| **Resources required** | Test cases require more resources for documentation and execution. | Fewer resources are required to write test scenarios. |
| **Inputs** | It includes all positive and negative inputs, expected results, navigation steps, etc. | They are one-liner statements. |
| **Time requirement** | It requires more time compared to test scenarios. | Test scenarios require less time. |
| **Maintenance** | They are hard to maintain. | They require less time to maintain. |

**When do we Write Test Cases?**

Test cases are written in different situations:

* **Before development:** Test cases could be written before the actual coding as that would help to identify the requirement of the product/software and carry out the test later when the product/software once gets developed.
* **After development:** Test cases are also written directly after coming up with a product/software or after developing the feature but before the launching of a product/software as needed to test the working of that particular feature.
* **During development:** Test cases are sometimes written during the development time, parallelly. so whenever a part of the module/software gets developed it gets tested as well.

So, test cases are written in such cases, as test cases help in further development and make sure that we are meeting all the needed requirements.

**Why Write Test Cases?**

Test cases are one of the most important aspects of software engineering, as they define the way in which the testing would be carried out. Test cases are carried out for a very simple reason, to check if the software actually works or not. There are many advantages of writing test cases:

* **To check whether the software meets customer expectations:**Test cases help to check if a particular module/software is meeting the specified requirement or not.
* **To check software consistency with conditions:**Test cases determine if a particular module/software work with a given set of conditions.
* **Narrow down software updates:**Test cases help to narrow down the software needs and required updates.
* **Better test coverage:**Test cases helps to make sure that all possible scenarios are covered and documented.
* **For consistency in test execution:**Test cases help to maintain consistency in test execution. A well-documented test case helps the tester to just have a look at the test case and start testing the application.
* **Helpful during maintenance:**Test cases are detailed which makes them helpful during the maintenance phase.

**Test Case Template**

Let’s look at a basic test case template for the login functionality.

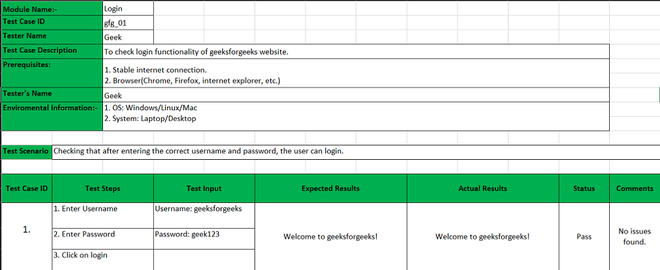
* The Test case template contains the header section which has a set of parameters that provides information about the test case such as the tester’s name, test case description, Prerequisite, etc.
* The body section contains the actual test case content, such as test Id, test steps, test input, expected result, etc.

Below is the table that shows the basic template of a test case:

| **Fields** | **Description** |
| --- | --- |
| **Test Case ID** | Each test case should have a unique ID. |
| **Test Case Description** | Each test case should have a proper description to let testers know what the test case is about. |
| **Pre-Conditions** | Conditions that are required to be satisfied before executing the test case. |
| **Test Steps** | Mention all test steps in detail and in the order to be executed from the end-user’s perspective. |
| **Test Data** | Test data could be used as input for the test cases. |
| **Expected Result** | The result expected after executing the test cases. |
| **Post Condition** | Conditions need to be fulfilled when the test cases was successfully executed. |
| **Actual Result** | The result which system shows once the test case was executed. |
| **Status** | Set status as Pass or Fail on the expected result against the actual result. |
| **Project Name** | Name of the project to which test case belongs to. |
| **Module Name** | Name of the module to which test case belongs to. |
| **Reference Document** | Mention the path of the reference document. |
| **Created By** | Name of the tester who created the test cases. |
| **Date of Creation** | Date of creation of test cases. |
| **Reviewed By** | Name of the tester who reviews the test case. |
| **Date of Review** | When the test cases were reviewed. |
| **Executed By** | Name of the tester who executed the test case. |
| **Date of Execution** | Date when the test cases were executed. |
| **Comments** | Include comments which helps the team to understand the test cases. |

In the given template below it’s clearly identifiable that the section from module name to test scenario is the header section while the table that lies below the test scenario (from test case id to comments) is the body of the test case template.

Here a test case template for login functionality has been created with its parameters and values.



*Test Case Template*

**Best Practice for Writing Test Case**

There are certain practices that one could follow while writing the test cases that would be considered beneficial.

1. **Simple and clear:**Test cases need to be very concise, clear, and transparent. They should be easy and simple to understand not only for oneself but for others as well.
2. **Maintain the uniqueness:**While writing the test cases, it’s necessary to make sure that they aren’t being written over and over again and that each case is different from the others.
3. **Zero Assumptions:**Test cases should not contain assumed data, and don’t come up with features/modules that don’t exist.
4. **Traceability:**Test cases should be traceable for future reference, so while writing it’s important to keep that in mind,
5. **Different input data:**While writing test cases, all types of data must be taken into consideration.
6. **Strong module name:**The module name should be self-explanatory while writing the test case.
7. **Minimal Description:**The description of a test case should be small, one or two lines are normally considered good practice, but it should give the basic overview properly.
8. **Maximum conditions:**All kinds of conditions should be taken into consideration while writing a test, increasing the effectiveness.
9. **Meeting requirements:**While writing the test case it’s important that the client/customer/end-user requirements are met.
10. **Repetitive Results:**The test case must be written in such a way that it should provide the same result.
11. **Different Techniques:**Sometimes testing all conditions might not be possible but using different testing with different test cases could help to check every aspect of a software.
12. **Create test cases with end user’s perspective:**Create test cases by keeping end-user in mind and the test cases must meet customer requirements.
13. **Use unique Test Case ID:**It is considered a good practice to use a unique Test Case ID for the test cases following a naming convention for better understanding.
14. **Add proper preconditions and postconditions:**Preconditions and postconditions for the test cases must be mentioned properly and clearly.
15. **Test cases should be reusable:**There are times when the developer updates the code, then the testers need to update the test cases to meet the changing requirements.
16. **Specify exact expected outcome:**Include the exact expected result, which tells us what will be result of a particular test step.

**Test Case Management Tools**

Test management tools help to manage the test cases. These tools are automated tools that decrease the time and effort of a tester as compared to the traditional way. Some test case management tools include advanced dashboards, easier bug, and progress tracking as well as management, custom test case templates, integration of test cases, inviting guest testers, managing team requirements and plans, and much more.

Below are some of the test case management tools:

* **Testpad:**Testpad is a simple tool that makes test case management easier. The software’s main motto says that it aims to find a bug that matters. A few features of Testpad include manual testing, reports of the test cases and software, dragging and dropping to make testing easier, inviting guest testers by email, building custom templates, and much more.
* **TestCaseLab:**TestCaseLab is easily manageable for the test cases and could swiftly integrate them with bug trackers. The features of TestCaseLab include Custom test cases, Test Runs, Integrations of test cases, Test Plans, tags and priority for test cases, search by name of test cases, description and tags, etc.
* **TestRail:**TestRail is another platform that aims to make test case management easier, it streamlines the software testing processes and along with more visibility into QA. The basic features of TestRail include management for test cases, plans, and runs, more test coverage, real-time insights into the QA progress, etc.
* **TestLodge:**TestLodge is a test case management tool that helps the entire team to manage their requirements, test plans, test cases, and test runs all in one single place and also with no user limit. The basic features of TestLodge include Test Plans, Test Runs, Dashboard, Test Suite, and many more.

**Formal and Informal Test Case**

* **Formal Test Cases:** Formal test cases are test cases that follows the basic test case format. It contains the test case parameters such as conditions, Id, Module name, etc. Formal Test cases have set input data and expected results, they are performed as per the given order of steps.
* **Informal Test Cases:** Informal test cases are test cases that don’t follow the basic test case format. In these, as the tests are performed the test case are written in real-time then pre-writing them and the input and expected results are not predefined as well.

**Types of Test Cases**

* **Functionality Test Case:**The functionality test case is to determine if the interface of the software works smoothly with the rest of the system and its users or not. Black box testing is used while checking for this test case, as we check everything externally and not internally for this test case.
* **Unit Test Case:**In unit test case is where the individual part or a single unit of the software is tested. Here each unit/ individual part is tested, and we create a different test case for each unit.
* **User Interface Test Case:**The UI test or user interface test is when every component of the UI that the user would come in contact with is tested. It is to test if the UI components requirement made by the user are fulfilled or not.
* **Integration Test Case: Integration** testing is when all the units of the software are combined together and then they are tested. It is to check that each component and its units work together without any issues.
* **Performance Test Case:**The performance test case helps to determine response time as well as the overall effectiveness of the system/software. It’s to see if the application will actually handle real-world expectations.
* **Database Test Case:**Also known as back-end testing or data testing checks that everything works fine with respect to the database. Testing cases for tables, schema, triggers, etc. are done.
* **Security Test Case:**The security test case helps to determine that the application restricts actions as well as permissions wherever necessary. Encryption and authentication are considered as main objectives with regard to the security test case. The security test case is done to protect and safeguard the data of the software.
* **Usability Test Case:**Also known as a user experience test case, it checks how user-friendly or easy to approach a software would be. Usability test cases are designed by the User experience team and performed by the testing team.
* **User Acceptance Test Case:**The user acceptance case is prepared by the testing team but the user/client does the testing and review if they work in the real-world environment.

**Example**

Below is an example of preparing various test cases for a login page with a username and password.

**Unit Test case:**Here we are only checking if the username validates at least for the length of eight characters.

| **Test Id** | **Test Condition** | **Test Steps** | **Test Input** | **Test Expected Result** | **Actual Result** | **Status** | **Remarks** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1. | Checks if the username field accepts the input of thirteen characters. | 1. Give input | geeksforgeeks | Accepts for thirteen characters. | Accepts for thirteen characters. | Pass | None |

Here it is only checked whether the passing of input of thirteen characters is valid or not. So since thirteen character word ‘geeksforgeeks’ is entered then the test is successful else it would have failed for any other test case.

**Functionality Test case:** Here it is checked whether the username and password both works together on the login click.

| **Test Id** | **Test Condition** | **Test Steps** | **Test Input** | **Test Expected Result** | **Actual Result** | **Status** | **Remarks** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1. | Check that with the correct username and password able to log in. | 1. Enter the username  2. Enter the password  3. Click on the login | username: geeksforgeeks  password: geeksforever | Login successful | Login successful | Pass | None |
| 2. | Check that if with an incorrect username and password able to not login. | 1. Enter the username  2. Enter the password  3. Click on the login | username: geeksforgeeks  password: geekstogether | Login unsuccessful | Login unsuccessful | Pass | None |

Here it is being checked whether passing wrong and right inputs if the login functionality is working or not, it’s showing login successful for the right credentials and unsuccessful for the wrong ones, hence both tests have passed otherwise would have failed.

**User Acceptance Test Case:**Here the user feedback is taken if the login page is loading properly or not.

| **Test Id** | **Test Condition** | **Test Steps** | **Test Input** | **Test Expected Result** | **Actual Result** | **Status** | **Remarks** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1. | Check if the loading page loading efficiently for the client. | 1. Click on login button. | None | Welcome to login page. | Welcome to login page. | Fail | The login page is not loaded due to a browser compatibility issue on the user’s side. |

Here it is being checked in by clicking on the login button if the page is loaded and the ‘Welcome to login page’ message is displayed. The test has failed here as the page was not loaded due to a browser compatibility issue, it would have loaded if the test would have passed.