HOW TO WRITE TEST CASES FOR SOFTWARE

A test case offers a way of validating whether a software component is free of bugs and performing as it should. Many suggest that the benefit of a strong test case is that it can guide any user, prompting them to walk through the test steps and execute the actions required for validation. And it's true that a test case should be able to do that.

At the same time, others purport that the objective of a test case is to automate steps to validate that a piece of software is free of bugs. After all, automating is the only way to rule out operator error.

In fact, there are several reasons automating test cases is preferable:

- Saves time and resources
- Increases test coverage
- Improves testing accuracy
- Simulates practically endless quantities of users
- Boosts the development team's morale as they are no longer stuck in repetitive actions

Regardless of your objective, it's important to understand the following best practices for writing a test case.

7 Tips for Writing Strong Test Cases

Anyone can write a test case but you want yours to be strong, clear and able to meet your objectives as described above. Here are seven things to keep in mind when getting ready to write a test case.

Don’t Over Complicate It

When writing test cases, less really is more. If you’re writing a test case for the first time, you may feel overwhelmed with questions. You may wonder if every test case requires relationships? Or you may ponder how much detail is too much?

Templates outline steps to take and certain requirements for writing a test case. But depending on your comfortability with the material and the objectives you may choose to leave out some elements that you do not deem necessary to your test. And that’s ok.

Focus on keeping the test simple and clear, and don’t over complicate it by imposing unnecessary standards.

Make it Reusable

It’s important that a strong test case can apply to many different scenarios in which software needs to be validated. The idea is that a test case should offer longevity and provide value to the entire team. Keeping this principle in mind will help to save time re-writing test cases in the future.

Look At it Critically

Does it perform the way you want it to?

When you’ve created a test case look at it critically. Not just as a test case writer, but as an end-user. Even if your test case will be automated, it’s still recommended that you look at it through the lens of the person who will be implementing the test.

Taking this fresh perspective allows you to identify any pitfalls or vulnerabilities in the testing model and make the necessary adjustments to address them.

Incorporate a Strong Title

Creating strong naming conventions may not always be the first thing that crosses your mind when it comes to writing a test case. However, the title should promote a strong understanding about what it is you are trying to validate. For instance, if you are testing the shopping cart feature of an application, include *Shopping Cart* along with test identification information in the title.

Include a Valuable Description

The title needs to be strong, but that doesn’t discount the importance of a good description. Using this space wisely means defining your test in the description and including details such as the test environment, test data and importantly, test assumptions.

Don’t Forget Assumptions

Including assumptions in your test helps ensure clarity. The same goes for preconditions that must be met before taking the test as well. Assumptions and preconditions include:

- Notating what page the test starts on
- Setup requirements, or any special requirements
- Dependencies in the testing environment

Use Clear and Concise Language

When it comes to writing test cases, the words “clear and concise” are used frequently. Use language that is simple and easy to understand, but which still meets the requirement of completeness in terms of the relevance of information included.

Utilize Testing Techniques

There are a few to choose from:

- **Boundary Value Analysis** (BVA) - The purpose of this technique is to test for errors at boundaries of a specified range.
- **Equivalence Partition** - This approach is generally used to reduce the total number of test cases by dividing the data range into groups.
- **Error Guessing** - As the name suggests, this informal methodology utilizes the tester’s experience to brainstorm the error-prone areas of the application.
- **State Transition** - This dynamic technique is utilized when software behavior changes from one state to the next.

How to Write A Test Case

For the savvy developer, there are many ways to write a test case.

Levels

The key characteristics of each test case describe an action, an input, event or anticipated response. Because of the broad description and the various ways a developer can attack this type of project, working with a test case in levels helps to avoid duplication of efforts.

Level 1: At this level, test cases are written specs that are available and any usable documentation. These test cases are in their most basic state.

Level 2: Level 2 test cases are programmed via a more practical approach which takes into consideration system flow of an application and how it functions.

Level 3: At Level 3, you begin grouping like test cases and writing test procedures. The latter is the name given to a group of up to 10 cases.

Level 4: Automation is the order of the day at this level where you focus on minimizing human interaction for best results.

Test Case Fields

It's expected that most test cases will contain roughly the same fields as follows:

Test Case ID

This is a unique identifier for each test case. Typically it's numeric.

Purpose

The purpose should not be longer than two sentences outlining the objective of the test case.

Prerequisite

This is where you include any conditions that must be met to run the test. This could include your assumptions and preconditions if you haven't already stated them somewhere else.

Test Data

These are elements used in a test case, including all variables and possible values, log in conditions and the like.

Test Steps

This is the place for detail. Include each step for completing the test from the end-user's perspective.

Expected Results

What does a bug-free outcome look like? Describe the desired results from running the test.

Actual Results

Record the test outcome.

Result

This is binary, pass or fail.

Comments

Include valuable information that will help the software developer debug.

Example Test Case

Title: Shopping Cart – Add items to cart

Description: A registered user should be able to successfully add an item or item(s) to cart.

Precondition: The user must already be registered using their email address and chosen password.

Assumption: The user must be on the mobile app.

Test Steps:

- Navigate to the iOS app on mobile device
- In the log-in field, enter the email of registered user
- Select 'Next'
- Type in the password of registered user
- Select 'Sign In'
- Choose Item from eCommerce Screen upon Sign In - Air Purifier
- Select 'Add to Cart'

Expected Result: Air Purifier item should be present in cart

Writing A Test Case Has Never Been Easier

Armed with the knowledge of how to write stronger test cases, you might wish to take things a step further by removing human error altogether. BMC offers a number of Application Deployment and DevOps Tools that help developers at all levels create new software, test and *automate the process*.

Let us help your DevOps team deliver applications quicker and with higher quality by providing self-service development capabilities that work within existing continuous delivery processes. For more information on how you can partner with BMC on testing and software development, contact us today.