Manual Testing - Series 01

How I Write Effective Test Cases: My Practical Approach

Writing test cases is an important part of software testing, and everyone has their own approach. There is no single right way to write test cases what matters is clarity, accuracy, and making them useful for testing. Over time, I have developed my own structured yet simple approach to writing test cases that work well for me and my projects.

In this blog, I will share my way of writing test cases, along with examples, best practices, and different types of test scenarios, including boundary testing, security testing, UI & usability testing, and browser-specific testing.

1. My Test Case Writing Approach

A good test case should be:

✅ Clear & Simple – The language should be easy to understand, even for someone new to the project.
✅ Prioritized – High, Medium, or Low priority, based on business and user impact.
✅ Step-by-Step – A well-structured test case makes execution easier and avoids confusion.
✅ Linked to Requirements – Each test case should relate to a feature or functionality.
✅ Independent – One test case should not depend on another test case.
✅ Well-Defined Expected & Actual Results – The expected outcome should be clear, and testers should record the actual result for comparison.

This approach ensures that test cases are easy to execute, reusable, and practical for real-world testing.

2. My Test Case Format (With Prioritization)

I organize my test cases in the following format:

Field	Description
Test Case ID	A unique number or code assigned to the test case.
Test Case Title	The features or functions are being tested.
Test Case Description	A short, clear description of what the test case covers.
Preconditions	Any setup needed before executing the test.
Test Steps	Step-by-step instructions for executing the test.
Test Data	Inputs required for the test (if applicable).
Priority	High, Medium, or Low, based on business impact.
Expected Result	What should happen if the test passes.
Actual Result	The real outcome after executing the test.
Status	Pass, Fail, Blocked, or Not Executed.

.

This format keeps everything organized clear and focused on execution.

3. Example Test Case: Login Functionality

Positive Scenario

Field	Details
Test Case ID	TC_001
Test Case Title	Login
Test Case Description	Verify login with valid credentials.
Preconditions	The user should be registered in the system.
Test Steps	1. Open the login page. 2. Enter a valid username and password. 3. Click the "Login" button.
Test Data	Username: test user Password: Test@123
Priority	High
Expected Result	Users should be redirected to the dashboard.
Actual Result	(To be updated after execution)
Status	(To be updated after execution)

Negative Scenario

Field	Details
Test Case ID	TC_002
Test Case Title	Login
Test Case Description	Verify login with invalid credentials.
Preconditions	The user should be registered in the system.
Test Steps	1. Open the login page. 2. Enter a valid username and password. 3. Click the "Login" button.
Test Data	Username: wrong user Password: Wrong@123
Priority	High
Expected Result	An error message should be displayed stating "Invalid username or password."
Actual Result	(To be updated after execution)
Status	(To be updated after execution)

This is my approach, and I find this format easy to follow, execute, and update.

4. Writing Test Cases for Different Scenarios

Every application has different testing needs. Here are some case examples for various scenarios:

◈ Boundary Test Cases (Checking limits and edge values)

Example: A password field allows 6 to 12 characters.

Test Data	Expected Result
5 characters	Error message displayed
6 characters	Password accepted
12 characters	Password accepted
13 characters	Error message displayed

◈ Security Test Cases (Ensuring system security)

Example: Testing SQL Injection in the login field.

Test Data	Expected Result
admin' OR '1'='1' --	Login should be denied and error logged.

◈ UI & Usability Test Cases (Checking visual design and user-friendliness)

Example: Checking button alignment and accessibility.

Test Data	Expected Result
Open login page	The “Login” button should be clearly visible and aligned properly.
Navigate using Tab key	The cursor should move in a logical order.

🔷 Browser-Specific Test Cases (Ensuring cross-browser compatibility)

Example: Checking login functionality on different browsers.

Browser	Expected Result
Chrome	Works correctly
Firefox	Works correctly
Safari	Works correctly

5. Best Practices for Writing Test Cases

✅ **Use simple language** – Avoid complex words or unnecessary details.
✅ **Write independent test cases** – Each test should be executable on its own.
✅ **Prioritize test cases** – Focus on critical functionality first.
✅ **Ensure complete coverage** – Write test cases for both positive and negative scenarios.
✅ **Use real-world test data** – Data should be practical and relevant to actual user inputs.
✅ **Include boundary value cases** – Always test input limits.
✅ **Document actual results** – Track whether the test passed or failed.
✅ **Review test cases regularly** – Keep them updated as the system evolves.

✅ **Step-by-Step** – A well-structured test case makes execution easier and avoids confusion.

6. Final Thoughts

This is **my personal approach** to writing test cases. It helps me stay organized, ensures complete test coverage, and makes execution smoother. Since every tester has their own style, I believe the best approach is one that works for you and your team.

💡 **What do you think? Do you have any suggestions to improve my approach? Let's discuss and learn together!** 😊