# **Standard Operating Procedure (SOP): Test Case Document Creation**

## **1. Purpose**

This SOP aims to establish a standard procedure for the creation, review, and management of test case documents. By following this procedure, we aim to ensure that:

* Test cases are **consistent, clear, and comprehensive**.
* The testing process can be conducted **efficiently and in a structured manner**.
* The quality of the software under test is **guaranteed** through adequate test coverage.
* It facilitates the **maintenance and reuse** of test cases in the future.

## **2. Scope**

This SOP applies to all QA team members responsible for designing and documenting test cases for any feature or application module to be tested.

## **3. Responsibilities**

* **QA Engineer:**
  + Understands feature requirements.
  + Designs and writes test cases according to standards.
  + Ensures test cases cover both positive and negative scenarios.
  + Prepares necessary test data.
  + Updates test cases if requirements change.
* **Sr. QA Engineer / QA Lead:**
  + Reviews test cases created by QA Engineers.
  + Ensures adequate test coverage and quality test cases.
  + Provides feedback and guidance.
  + Approves test cases before execution.
* **Product Manager / Business Analyst:**
  + Provides clear and detailed feature requirements.
  + Offers clarification regarding requirements when needed by the QA team.
  + Participates in test case reviews (optional, to ensure test cases align with business expectations).

## **4. Definitions**

* **Test Case:** A set of conditions, steps, and expected results used to verify a specific functionality of a software application.
* **Requirement Document (PRD/SRS):** A document that describes the functionalities, features, and constraints of the system to be developed.
* **Test Data:** Data used as input during test case execution.
* **Test Case ID:** A unique identifier for each test case.
* **Module:** A functional part or area of the application.
* **Pre-conditions:** Conditions that must be met before a test case can be executed.
* **Expected Result:** The anticipated outcome of executing the test case steps.
* **Actual Result:** The observed outcome after executing the test case steps.

## **5. Test Case Document Creation Procedure**

### **5.1. Initial Preparation**

1. **Understand Requirements:**
   * The QA Engineer must thoroughly read and understand all relevant Requirement Documents (PRD/SRS, User Stories, Figma designs) for the feature to be tested.
   * Identify happy paths, alternative paths, and error conditions.
   * If there are ambiguities or questions, discuss them with the Product Manager/Business Analyst for clarification.
2. **Identify Feature/Module:** Clearly define which feature or module will be covered by these test cases.
3. **Access Template:** Use the standard test case document template provided (such as the example I provided previously).

### **5.2. Test Case Writing**

1. **Assign Test Case ID:**
   * Assign a unique ID to each test case using a consistent format (e.g., TC\_MODULE\_XXX like TC\_LOGIN\_001, TC\_FP\_001).
2. **Determine Module:** Fill in the "Module" column with the name of the main module or functionality relevant to this test case (e.g., Login, Forgot Password, Profile).
3. **Write Test Case Title:**
   * Create a concise, descriptive, and easy-to-understand title.
   * The title should reflect the main objective of the test case (e.g., "Successful Login with Valid Username").
4. **Define Pre-conditions:**
   * List all conditions that must be met before the test case can be executed.
   * Examples: "User is registered and has a valid Username & Password", "Account is locked".
5. **Write Test Steps:**
   * Write steps sequentially, clearly, and specifically.
   * Each step should be an actionable instruction (e.g., "1. Open the Login page.", "2. Enter a valid Username.").
   * Avoid ambiguous steps.
   * Include specific input data if relevant.
6. **Define Expected Result:**
   * Describe the anticipated outcome of executing the steps accurately and measurably.
   * There must be clear pass/fail criteria.
   * Examples: "User is successfully logged in and redirected to the homepage/dashboard.", "System displays an error message 'Username not found'."
7. **Testing Scenarios:**
   * Ensure test cases are created for positive scenarios (happy paths).
   * Also create test cases for negative scenarios (invalid input, error conditions, edge cases).
   * Consider boundary testing (e.g., minimum/maximum values).

### **5.3. Test Data Preparation**

1. Identify specific data required for each test case.
2. Ensure test data is available in the relevant testing environment (staging, UAT).
3. Document the test data used within the test case or in a separate document if extensive.

### **5.4. Review Process**

1. Once test cases are written by the QA Engineer, the Sr. QA Engineer or QA Lead will review the document.
2. **Review Focus:**
   * **Requirement Alignment**: Are all requirements covered?
   * **Clarity & Precision**: Are the test cases easy to understand and are the steps clear?
   * **Completeness**: Are any scenarios missed (positive, negative, edge cases)?
   * **Testability**: Can the test cases be executed and do they have clear expected results?
   * **Format Consistency**: Is the format and naming consistent with team standards?
3. Provide feedback and suggestions for improvement to the QA Engineer.
4. The QA Engineer revises based on feedback.
5. After revision, the Sr. QA Engineer / QA Lead provides approval for the test case document.

### **5.5. Management and Updates**

1. Store the test case document in a test case management system (e.g., Zephyr, TestRail) or a document repository (e.g., Confluence, shared Google Drive folder) with clear version control.
2. When there are changes to feature requirements or bugs are found that require new/updated test cases, the test case document must be revised and go through the re-review process.
3. Ensure that irrelevant test cases are marked as deprecated or removed if no longer in use.

## **6. Test Case Status Definition**

* **Pass:** The **Actual Result** precisely matches the **Expected Result** for the given test case steps. The functionality works as intended.
* **Fail:** The **Actual Result** deviates from the **Expected Result** for the given test case steps. The functionality does not work as intended, and a defect (bug) should be logged.
* **Blocked:** The test case cannot be executed due to an impeding issue (e.g., environmental issue, dependency bug, or missing test data). This requires resolution of the blocking issue before re-attempting execution.
* **Skipped:** The test case was intentionally not executed, usually due to a specific reason (e.g., functionality is out of scope for the current sprint, known deferral of the bug, or a duplicate test). A comment explaining the reason should be provided.
* **Not Run:** The test case has not yet been executed. This is the default status before execution.