**What is a Test Case**

Test case - is the smallest unit of the testing plan - which includes a description of necessary actions and parameters to achieve and verify the expected behaviour of a particular function or the part of the tested software.  
If you have a task to check some functionality, you can create a test script or user story. So you need to understand where to start testing, which general steps need to be executed and what the result should be.  
And then this scenario is broken down into more detailed parts – test cases – to define all positive, negative, localisation and other behaviours of the software.  
For example, testers need to test the functionality of uploading photos.

**We create a test scenario as:**

1. The user must be logged
2. Move to the "upload photos" page
3. Click the "upload" button
4. Select photos
5. Upload them

**Now, this scenario should be divided into detailed test cases, for example:**

* Check the logged user possibility to go to the "upload photos" page
* Check the not logged user possibility to go to the "upload photos" page
* Check whether the user can click "upload" button
* Is it opens a form to select a photo and possibility to close it
* What happens if you do not select photos, choose another file format (for example video), choose photos of a maximum size and so on
* Check the possibility to upload photos
* Check if photo is saved
* Possibility to reload or delete photos
* What happens with photos in the case of the disappearance of the Internet or the device is switched off
* Are all buttons displayed correctly at another location or on different operating systems (if any difference)

And so on. The number of test cases depends on the experience and imagination of the tester.

Therefore, the process of writing test cases starts from forming a test scenario or user story, and then it can be divided to check different occasions.

**A structure of a Test Case**

The goal of Test Case documentation is to specify and communicate the specific conditions which need to be validated to enable an assessment of the system. Test Cases are motivated by many things but will usually include a subset of Use Cases, performance characteristics and risks in the project.

A good test case template maintains the test artifact consistency for the test team and makes it easy for all stakeholders to understand the test cases. Writing of test cases in a standard format reduce the test efforts and the error rate. Test cases format is more desirable in case if you are reviewing test cases from experts.

|  |  |
| --- | --- |
| **Test Case Field** | **Description** |
| **Test case ID** | Each test case should have a unique ID. |
| **Test Priority** | It is useful while executing the test.   * Low * Medium * High |
| **Test Designed by** | Tester's Name |
| **Date of test designed** | Date when test was designed |
| **Test Executed by** | Who executed the test(tester) |
| **Date of the Test Execution** | Date when test needs to be executed |
| **Name or Test Title** | The title should provide a concise, revealing description of the test case, such as “ Reset Pass ”. The title is important because it’s often the first or only thing you see when you are scanning a list of test cases. Clear titles are the key to help testers to find quickly the right test cases. |
| **Description/Summary of Test** | A detailed description of the test case. In this section, you can also set up categories to organize your test cases into logical groups. |
| **Pre-condition** | Any requirement that needs to be done before execution of this test case |
| **Test Steps** | Test Steps section gives the tester a numbered list of the steps to perform in the system, which makes it easier to understand the test case.  It is recommended to have 3-8 test steps per one test case. Too many steps make it difficult for developers and testers to reproduce the steps when a bug report is filed against the test case. |
| **Test Data** | You can enter test data directly in the test data field, or refer to a separate file that contains test data for one or more test cases. By using a test data file, you avoid hard coding test data in the test case, so a single test case can be used to test several sets of test data. |
| **Expected Results** | Mention the expected result including error or message that should appear on the screen. The tester needs to know the expected result in order to assess whether the test case is successful. The optimal level of detail in this field varies from situation to situation. |
| **Post-Condition** | What would be the state of the system after running the test case? |
| **Status (Fail/Pass)** | Mark this field as failed, if actual result is not the same as the expected result |
| **Notes/Comments/Questions:** | If there are some special conditions which is left in the above field |
| **Requirements** | List of the requirements for a particular test cycle. |
| **Attachments/References** | The files and documents that are attached to the test case, such as screen captures and other supporting material. |
| **Automation? (Yes/No)** | Fill “YES” when test cases are automated |

**Types of test cases**

At the beginning of the career, any tester faced with the problem when a team lead, project manager or client expresses his dissatisfaction with the fact that you wrote a few test cases.

In order to efficiently cover the functional by tests, test cases need to be divided into types. If you start doing it, then their number will increase at least in three times. Various sources describe types in different ways, but the essence of the division does not change. We offer the following types of test cases that should divide your test plan:

**Positive**

There are test cases aimed at checking the correct operation of the claimed functionality using the correct input format specified in the software documentation.

**For example, positive test cases check all right formats of emails, which must meet the following requirements.**

I. The first part of the email address, before @ may contain any of these characters ASCII:

* latin letters, regardless of the case - from a to z
* numbers from 0 to 9
* the special characters # $% & '\* + - / = ^ \_ `{|} ~ !?
* point "." but if it is among the other characters
* space and characters "(): <> @ [\] allowed with restrictions for a comment or indication of the name, etc.

II. Domain part - after the @ symbol may contain:

* latin letters, regardless of the case - from a to z
* numbers from 0 to 9, if the domain name contains not only the numerical values
* and "-" if it is between other characters

**Negative**

There are test cases that check your anticipated every possible situation that should lead to an error message. Also, this type of test cases includes a verification that can lead to unexpected situations, ie those that are not described in the documentation.

For example, you can test the field email, introducing the characters that are not included in the list mentioned above. You can also try to interrupt the fields, check whether the data is stored in the system reboot or exposure to other external factors.

**Boundary value**

To check the values on either side constraints. One of these relates to tests positive, the other to negative. It is better to isolate them not to miss. These tests are an indication that you own test design, which you can see below.  
For example, you found the information in the documentation that the password must contain at least 6 and no more than 60 characters. So you have to ascertain what happens if you type 5, 6, 60 and 61 characters. Do not forget about a case when the field is empty.  
If documentation does not describe such restrictions, you can offer them themselves, discussing with the team!

**Integration**

Check connections between different parts of the program. This is not exactly the type of test cases, but rather the level of testing. But such tests are required. You have to describe them, especially if your system consists of at least two modules.  
You can write test cases to check the appearance of the data entered in another part of the software.  
For example, if you have a payment for a certain type of functionality. Then you definitely need to ascertain whether that functionality becomes available after payment. After all, developers are likely to have implemented these parts separately, and problems could arise when they integrate those parts.

**Testing localisation**

Check all UI elements in different languages and their locations (if there is a support for languages with different rules of writing and reading).  
For example, if your software supports one of the location where the UI is placed from the right to left, you should pay attention to the work of Drop-Down List, check boxes, switching elements On / Off, etc.

Written tests to check GUI. You can describe the appearance of tips in the program hotkeys, errors, etc.  
If you have enough time, you can write test cases that will help you with cross-platform testing, especially if the program depends on platforms.  
If you have a great software that supports multiple languages, make a separate chapter for localisation test case.

If you are not using any test case management tool, you can use any open source tool or Excel Sheet to manage and execute your test cases.

Test case templates and examples are very useful because using them you can save time and resources for the cover product by a large number of test cases.

Test case formats vary by organisation. There are a lot of methods of the test case documentation, some of them:

**Example 1**

It is very convenient in case when the tester needs to record great detail of each step. Well suited to the case when test cases are made for new testers. It will help them to cover product by quality tests and do not miss any important data.

|  |  |
| --- | --- |
| **Project Name:**Banking System | |
| **Test Case** | |
| **Test Case ID:** **BU\_001** | **Test Designed by:**<Name> |
| **Test Priority (Low/Medium/High):**Med | **Test Designed date:**<Date> |
| **Module Name:**Bank login screen | **Test Executed by:**<Name> |
| **Test Title:** Test the Login Functionality in Banking | **Test Execution date:**<Date> |
| **Description:** Verify login with valid username and password |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Pre-conditions:**User has valid username and password  **Dependencies:** | | | | | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** | |
| 1 | Navigate to login page |  | User should be able to login | User should is able to login | Pass |  | |
| 2 | Provide valid username | User= [example@gmail.com](mailto:example@gmail.com) | Credential can be entered | As Expected | Pass |  | |
| 3 | Provide valid password | Password: 1234 | Credential can be entered | As Expected | Pass |  | |
| 4 | Click on Login button |  | User logged | User logged successfully | Pass |  | |

|  |
| --- |
| **Post-conditions:** User is validated with database and successfully login to the account. The account session details are logged in the database. |

**Example 2**

If in unnecessary details, this example saves time and resources.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | BU\_001 | **Test Case Description** | Test the Login Functionality in Banking | | |
| **Created By** | <Name> | **Reviewed By** | <Name> | **Version** | 2.1 |
| **QA Tester’s Log** | | | Review comments from Bill incorporate in version 2.1 | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tester's Name** | <Name> | **Date Tested** | 1-Jan-2017 | **Test Case (Pass/Fail/Not Executed)** | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S#** | **Pre-Conditions:** | | **S#** | **Test Data** |
| 1 | Access to Chrome Browser | | 1 | Userid = mg12345 |
| 2 |  | | 2 | Pass = df12@434c |
| 3 |  | | 3 |  |
| 4 |  | | 4 |  |
| **Test Scenario** | | Verify on entering valid userid and password, the customer can login | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step #** | **Step Details** | **Expected Results** | **Actual Results** | **Pass / Fail / Not executed / Suspended** |
| 1 | Navigate to http://Banksite.com | Site should open | As Expected | Pass |
| 2 | Enter Userid & Password | Credential can be entered | As Expected | Pass |
| 3 | Click Submit | Cutomer is logged in | As Expected | Pass |
| 4 |  |  |  |  |

**Example 3**

If necessary, accurate test data to be tested, it will be convenient to use. Which has “Data Set” for different variations in data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | TC\_Functionality\_01 | | | | |
| **Priority** | | High | | | | |
| **Dercription** | | Test the Login Functionality in BAnking | | | | |
| **Module** | | Main login screen | | | | |
| **Prepared by** | | <Name> | **Date Prepared** | | 1-Jan-2017 | |
| **Tested by** | | <Name> | **Date Tested** | | 13-Jan-2017 | |
| **Test Activities** | | | | | | |
| **No** | **Step Description** | | | **Expected Result** | | **Actual Result** |
| 1 | Navigate to http://BankSite.com | | | Site should open | | As expected |
| 2 | Enter Login & Pasword | | |  | |  |
| 3 |  | | |  | |  |
| **Test Data Sets** | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Type** | **Data Set 1** | **Data Set 2** | **Data Set 3** |
| Login or Email | User1 | example@gmail.com | User2 |
| Password | 123456 | 123456 | qwerty@!$ |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| **Test Case Result** | | Pass | |

**Example 4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Summary** | **Steps** | **Expected Result** | **Actual Result** | **Notes** |
| **1** | **Upload a photo as not logged user** | | | | |
|  | Pre-steps | 1.Open application |  |  |  |
| 1.1 | Check if the Upload a photo page opens photo page opens | 1.Click the Upload a photo link | The Upload a photo page doesn’t open the page doesn’t open | Fail | Bug#1 |
| ... | ... | ... | ... | ... | ... |
| 2 | **Upload a photo as logged user** | | | | |
|  | Pre-steps | 1.Open application Execute login with correct data |  |  |  |
| 2.1 | Check if the Upload a photo page opens photo page opens | 1. Click the Upload a photo link | The Upload a photo page opens | Pass |  |
| 2.2 | Check if the user can go back | 1. Click the Upload a photo link 2. Click the Back button | Previous page opens |  |  |
| 2.3 | Check if the Upload button works | 1. Click the Upload a photo link 2. Click on the Upload button | The form for choosing a photo opens | Pass |  |
| ... | ... | ... | ... | ... | ... |