A software product is like an airplane: it must undergo a technical check before launch.

Quality Assurance is a necessary step towards launching a successful software product. It is just a small part of all the project work, but nobody said it would be simple.

There are so many types of software testing – automated and manual, exploratory and functional, compatibility, UI/UX, regression, unit, API, and performance testing. Good luck wrapping your head around all of them!

What is common for all these types, however, is that each requires you to write thorough QA testing documentation. The quality of test documents defines whether your work will prove useful or go in vain.

I've written this article to make your life a bit easier. So here it is, your ultimate guide on how to write software QA documentation that will work.

Make a Test Plan and a Test Progress Report

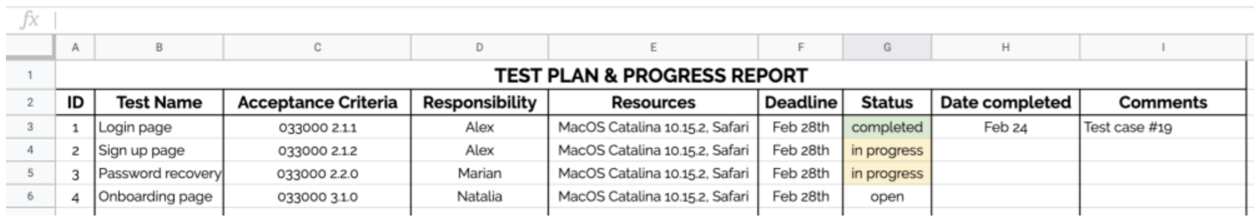
The test plan is a guiding document which outlines the bigger picture of the QA process, and includes a to-do list, strategy, resources, and schedule.

Before creating a QA plan document, ask yourself “What is the purpose of the software solution?” and “What features need to be tested?”. Do not rush into testing every single part of your software. You need to decide what methodologies, technologies, and tools you will use.

The test plan will help you understand the following:

- What are the acceptance criteria?
- What resources do you need? What operating systems, how many copies, and with what license? Do you need any technical consultants?
- Are your roles and responsibilities well-designated?
- What are testing timeframes?

The test progress report is another part of the QA documentation, which is similar to the test plan but with added data on the current progress. This document lets you and your development team monitor project progress and identify organizational issues, if any.



TEST PLAN & PROGRESS REPORT

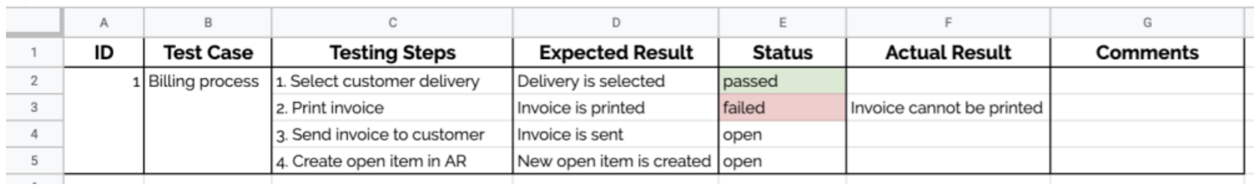
ID	Test Name	Acceptance Criteria	Responsibility	Resources	Deadline	Status	Date completed	Comments
1	Login page	033000 2.1.1	Alex	MacOS Catalina 10.15.2, Safari	Feb 28th	completed	Feb 24	Test case #19
2	Sign up page	033000 2.1.2	Alex	MacOS Catalina 10.15.2, Safari	Feb 28th	in progress		
3	Password recovery	033000 2.2.0	Marian	MacOS Catalina 10.15.2, Safari	Feb 28th	in progress		
4	Onboarding page	033000 3.1.0	Natalia	MacOS Catalina 10.15.2, Safari	Feb 28th	open		

Create Test Cases

Once you've clarified the set of functions that need to be tested in your test plan, you need to create a test case for each part of your software.

Test cases are pretty simple – this QA documentation consists of 7 sections: ID, Test Case, Testing Steps, Expected Result, Status, Actual Result, and Comments.

1. **ID** is a unique number assigned to your test case.
2. In the **Test Case** section, you point out the requirement(s) you will be testing and provide a link to it in the specifications document.
3. In the **Testing Steps** section, you list all the actions needed to complete a test case.
4. In the **Expected Result** section, you summarize the outcome of a particular test if it is successful.
5. In the **Status** section, you indicate if a particular step passed or failed testing.
6. In the **Actual Result** section, you explain the outcome of a failed test.
7. The **Comments** section is not obligatory, but you can add it to leave some additional notes.

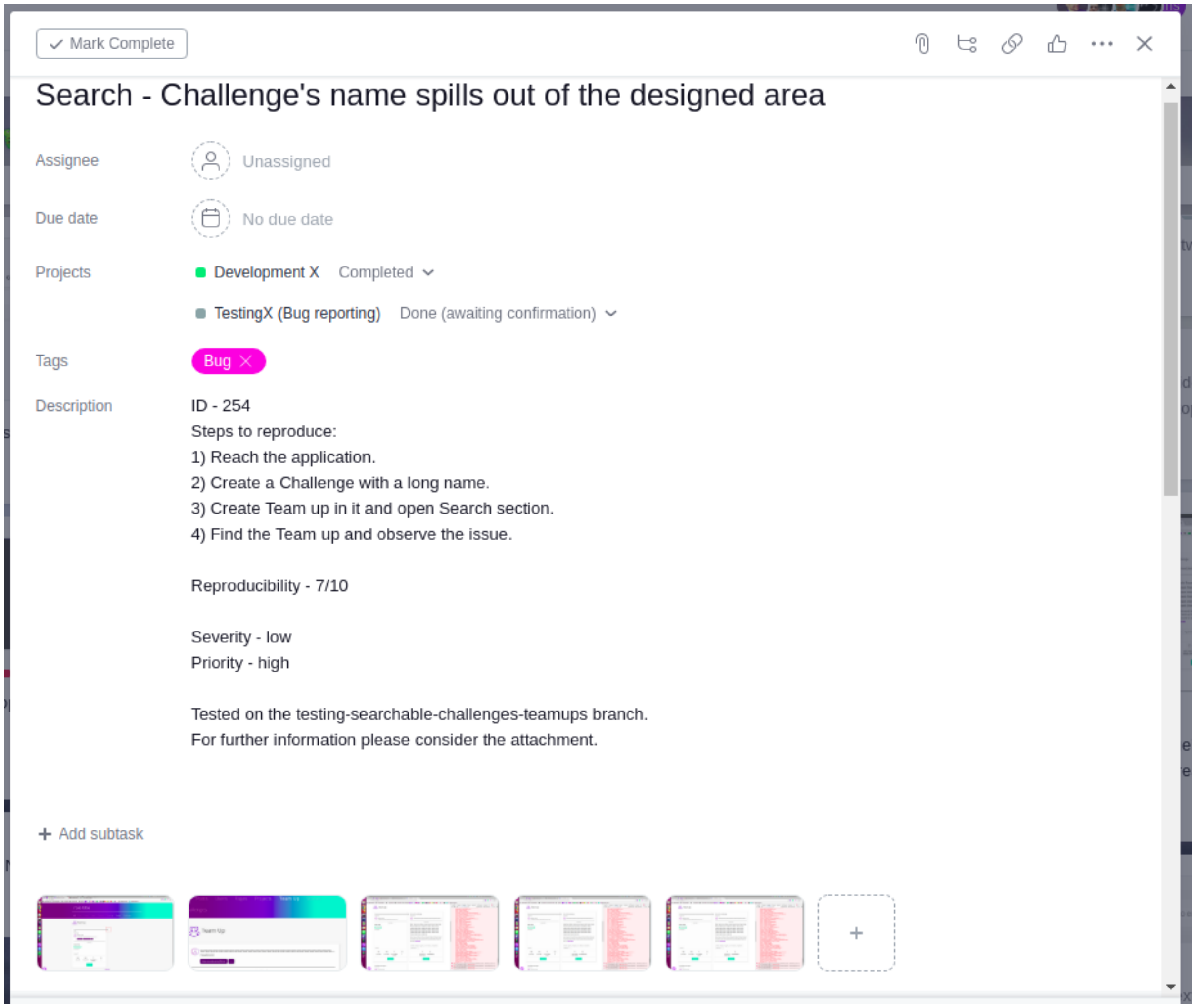


ID	Test Case	Testing Steps	Expected Result	Status	Actual Result	Comments
1	Billing process	1. Select customer delivery	Delivery is selected	passed		
		2. Print invoice	Invoice is printed	failed	Invoice cannot be printed	
		3. Send invoice to customer	Invoice is sent	open		
		4. Create open item in AR	New open item is created	open		

Write a Defect Report

The defect report is an important element of QA documentation. It registers any unwanted issue with your program. It is a crucial element of the project documentation, which navigates you towards getting a bug-free software solution.

Sounds simple, right? Yes, but only until you start documenting. Here, you can see an example of a typical defect report:



Mark Complete

Search - Challenge's name spills out of the designed area

Assignee	Unassigned
Due date	No due date
Projects	Development X Completed
	TestingX (Bug reporting) Done (awaiting confirmation)
Tags	Bug
Description	ID - 254

Steps to reproduce:
1) Reach the application.
2) Create a Challenge with a long name.
3) Create Team up in it and open Search section.
4) Find the Team up and observe the issue.

Reproducibility - 7/10

Severity - low
Priority - high

Tested on the testing-searchable-challenges-teamups branch.
For further information please consider the attachment.

The defect report consists of the following sections: identifier, summary, description, steps to reproduce, reproducibility, severity, priority, environment, and attachments.

1. Each particular software issue is assigned a unique number – the **identifier**. It optimizes navigation through QA documents and facilitates communication between developers, testers, and PMs.

2. In the **summary** section, you provide a concise answer to three simple questions: what happened, where, and under what circumstances.
3. In the **description** section**,** you describe the bug in detail. You should tell about the actual results and the expected ones. It's useful to provide a link to your software requirements.
4. Then, you write about the **steps to reproduce (STR)**. This shows developers exactly how to reproduce the issue. Don't miss a step or your report may return to you.
5. In the **reproducibility** section, you clarify if the bug appears every time you follow the STR. You should use numbers to show approximate chances, for example 7 times out of 10.
6. In the **severity** section, you explain how much harm the bug may bring to the project. In other words, it shows the technological degree of influence of the defect on the entire system. Even a small issue may badly affect the entire application.
7. **Priority** shows how important a particular defect report is. Priority can be denoted with the help of letters – “A” for the

most urgent and “Z” for the least urgent, numbers – “1” for the most urgent and “9” for the least urgent, or simply words like “high”, “medium”, or “low”.

8. In the **environment** section, you should define which operating systems or browser versions were affected.
9. Finally, the **attachments** include the list of videos, screenshots, or console logs files added to the defect report.

Keep These Useful Tips for Defect Report Writing in Mind

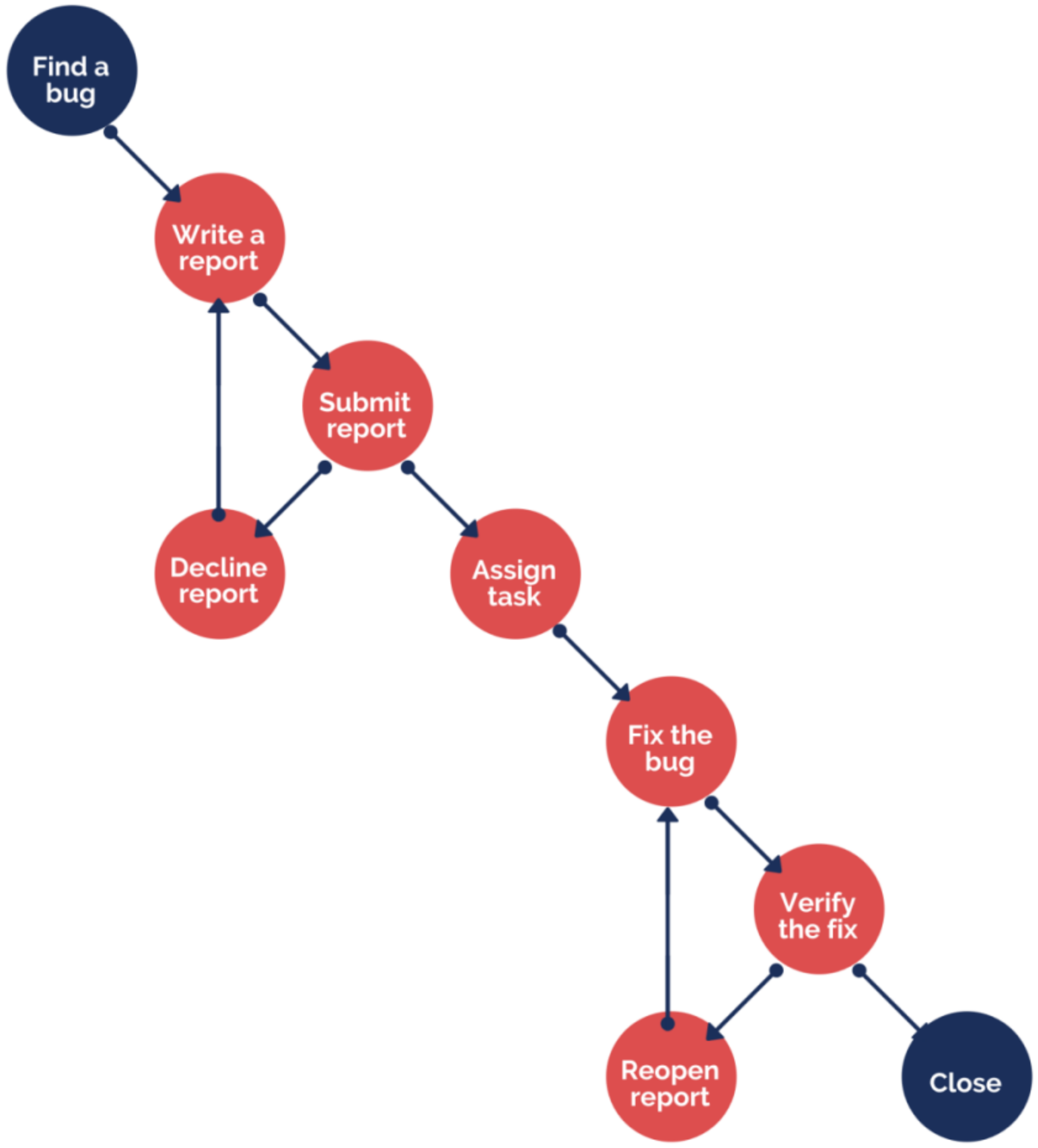
1. Write a sufficient and adequate summary. It does not matter if it is long or short. What matters is that it should be clear.
2. Have a look at the summary and the description. Do they look pretty much the same? You must have forgotten to outline expected and actual results in the description and to add the link to requirements.
3. Capture the issue with the help of a screenshot. It may save you and the development team a lot of time. Sometimes, one glance at the picture is just enough to understand the situation.

4. Before reporting the issue, try to reproduce it at least 3 times to be sure that it exists.
5. Report the issue ASAP and notify your project manager or product owner if the issue is major.
6. Check for grammar mistakes in your QA documentation so you're not taken down by the grammar police.
7. However comical it sounds, make sure that the issue is not a feature – review the documentation again!
8. Do not miss any important information in your Steps to Reproduce.

Submit a Defect Report

Defect reports go through a lifecycle – from the moment you report an issue to the moment the issue is closed.

Defect Report Lifecycle



The first step is to compile and **submit** the defect report. At this point, the Project Manager or a tech lead decides whether to **assign** it to a developer or to **decline** it on the grounds of insufficiency or inadequacy.

After the assigned developer has **fixed** the bug, you as a QA have to double-check and **verify** it has been fixed. If yes, you can **close** the defect report. The best practice is to close it in a week or two.

If the bug is not fixed, you **reopen** the defect report and send it back to the assigned developer. The bug-fixing process can be a long one, but you have to stay strong until all the defect reports are finally closed.

To Wrap Up

Quality Assurance is a process you simply cannot avoid. Each airplane prior to departure undergoes a technical check. If there is any issue, the aircraft is grounded until the problem is solved.

Similarly, each software product needs to be checked before launch. You cannot afford to deploy buggy software because users will not give your app a second chance.

Do you need to improve the quality of your software?

My company KeenEthics provides solid development and quality assurance services. In case you need an estimate for a similar project, feel free to get in touch.

If you have enjoyed the article, you should continue with What Is Prototyping and Why Do We Need It and Minimum Viable Product: Between an Idea and the Product.

P.S.

The original article posted on KeenEthics blog can be found here: How to Write QA Documentation That Will Work?