**How to create a test plan for software testing**

## What is a software test plan?

A test plan is a document that sets out the scope, approach, and schedule of intended testing activities. The test plan may also list the resources the software tester needs to function effectively.

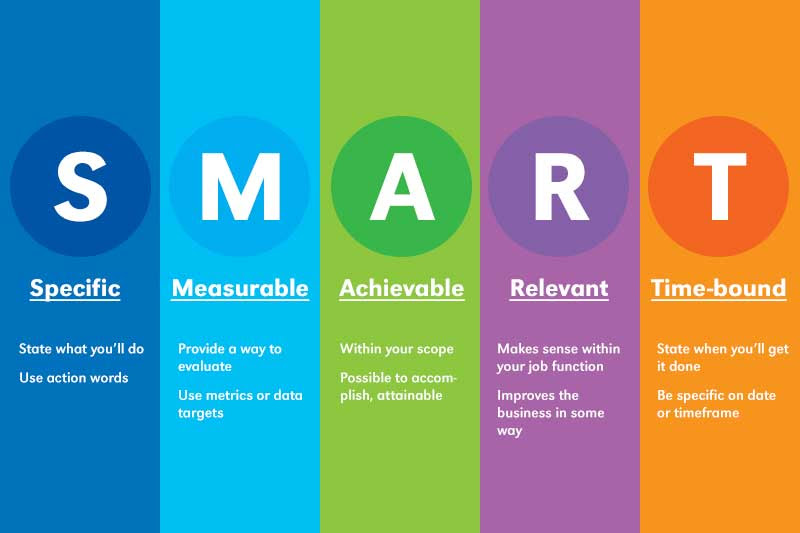
The test plan usually includes the following information:

1. The overall objective of the testing effort.  
2. A detailed outline of how testing will be conducted (the test approach).  
3. The features, applications, or components to be tested.  
4. Detailed scheduling and resource allocation plans for testers and developers throughout all stages of testing.

## What are the objectives of a software test plan?

The primary objective for a test plan is to produce documentation that describes how the tester will verify that the system works as intended. The document should describe what needs to be tested, how it will be tested, and who’s responsible for doing so.

By writing up a test plan, all team members can work in unison and communicate their roles to one another. You should consider creating some SMART objectives for your test plan.



[Image source](https://ghcc.org/en/5-steps-to-set-smart-objectives-examples/)

## What is a test case?

A test case is documentation created by the software tester that contains detailed information on what the test should accomplish. It’s an essential part of recording information about testing activities and results.

Test cases are used in conjunction with test plans. A test case should include the following information.

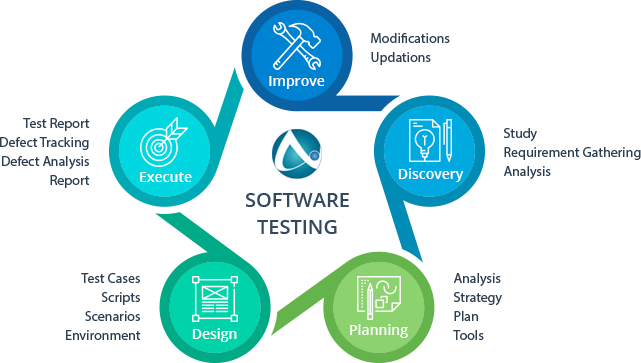
1. A unique name or number to identify it.  
2. The features, applications, or components covered by the test case.  
3. Specific data values required for input fields and button controls to be tested.  
4. The predicted results of actions taken during testing (the expected outcome).  
5. A description of the actual results following each action taken during testing (the actual outcome).  
6. An indication of whether or not the test case was successful.  
7. Any errors discovered.

## What’s the importance of a test plan?

A test plan is the foundation of every testing effort. It helps set out how the software will be checked, what specifically will be tested, and who will be performing the test. By creating a clear test plan all team members can follow, everyone can work together effectively.

Whether you’re [building an app](https://www.workpuls.com/blog/6-crucial-questions-startups-consider-building-an-app) or developing open-source software, a test plan is essential to delivering the final result.

A high-quality plan helps to identify risk areas, determine the order of testing activities, and allocate resources efficiently. The test plan becomes a useful reference document that can be referred back to throughout the product's development cycle.



[Image source](https://u-tor.com/topic/importance-of-testing-in-sdlc)

## How to keep the audience in mind when creating a test plan

Before you begin creating your test plan, you’ll need to identify your intended consumers and make sure their needs are being met. This will improve the quality of your test plan tenfold.

Here are the main things to ensure your test plan is:

* **Concise.** Your test plan should be no longer than one page with bullet points.
* **Organized.** Make sure all the information is logically grouped.
* **Readable.** The document should be easy to read, avoiding technical language where possible.
* **Flexible.** Your test plan should be adaptable and not fixed in stone. You want to create documentation that won't hold you back if new information comes along or changes need to be made.
* **Accurate.** Make sure all the information you've included is accurate.

## How to write a test plan

This might be the first job on your [software developer CV](https://www.wearedevelopers.com/magazine/quick-guide-how-to-write-a-software-developer-cv), and if that’s the case, you may need a cheat sheet to successfully write your initial test plan.

Luckily, we have you covered. This section will provide you with 14 essential things to include in your software test plan as part of the [QA process](https://www.globalapptesting.com/blog/qa-process).

### 1) Learn about the software

Before testing starts, it's important to learn everything you can about the software. Ask questions about how it was developed to learn about its intended purpose, how it works, and to garner information that might help you understand its functionality.

By understanding your software properly, you can create test cases that are relevant and useful for testing your product.

### 2) Define the scope of testing

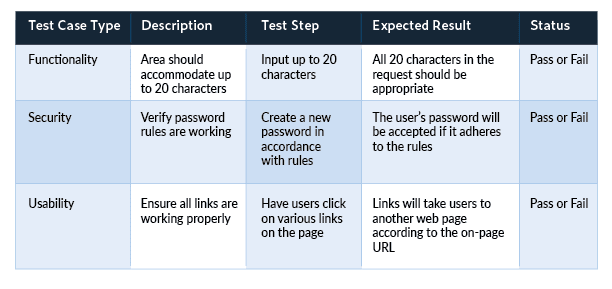
There’s no point in creating testing documents that are longer than the product itself. Before anything else, establish what exactly will be tested during the process, which modules or functions need to be covered in detail, and any other essential aspects you should know about.

### 3) Create test cases

One of the main tasks when developing a software testing document is creating test cases. A test case is a document that describes the steps taken to carry out your testing. It should include information such as:

* What needs to be tested
* How it will be tested
* Who will do the testing
* Expected results

 Here’s a simple spreadsheet for setting up test cases:



[Image source](https://www.parasoft.com/blog/how-to-write-test-cases-for-software-examples-tutorial/)

### 4) Develop a test strategy

The test strategy defines how you plan to implement testing. Your testers should all be working off the same game plan, so make sure every member of the team is aware of what they're supposed to be doing at any given time.

### 5) Define the test objective

Each test case should be linked to a test objective. The objective ensures every action is relevant and contributes toward making your software more valuable for customers. Test objectives can include things like:

* Testing known features
* Testing newly implemented features
* Performing exploratory tests
* Ensuring stability throughout the product lifecycle

### 6) Choose testing tools

You'll need to make sure you have the right [software testing solution](https://www.globalapptesting.com/blog/software-testing) to perform your testing activities. Some of these tools might be software-based, while others may require physical resources like test machines. It's important to choose appropriate tools for each specific job and not to rely on a one-size-fits-all solution.

### 7) Find bugs early

Leave time in your planning document for 'bug fixing' sessions. These allow you to identify problems with the software early on before they become too problematic or expensive to fix. This makes them easier and cheaper to tackle. Check out any [app security measures](https://dzone.com/articles/security-measures-for-open-source-based-apps), use every feature, and seek out what doesn't work well.

### 8) Define your test criteria

This should be part of the test case, but it's good to break it down separately. Test criteria are essentially your objectives broken down into smaller parts. They include specific information about how each objective will be met, which helps you track your testing progress.

Suspension criteria are criteria that need to be met before testing can stop. For example, you may want to suspend testing if a certain number of bugs have been found or if the software is unable to run due to performance issues.

Exit criteria are criteria that need to be met before testing can finish. For example, the test case should finish once each objective has been met and all bugs have been resolved.

### 9) Resource planning

In your software testing document, include a resource plan that lists the number of people required for the testing process. This should detail what each person's role is and any training they'll require to fulfill it effectively.



[Image source](https://stafiz.com/importance-of-resource-planning-in-project-management/)

### 10) Plan your test environment

In your test plan, include information about the environment where testing will take place, such as:

* Test hardware required for product testing.
* Sizing requirements for software and servers.
* Platforms supported by the product.
* Other essential information related to the environment that might affect your testing process.

### 11) Plan test team logistics

Test management is one of the most important parts of [implementing process](https://processbliss.com/three-tips-for-successfully-implementing-process-in-your-business/). If you're not able to communicate with your testers effectively, their progress will suffer and your testing document won't be as useful as it could be.

### 12) Schedule & estimation

In your test plan, include a schedule that allows you to outline specific testing milestones and deadlines. Milestones may include the initial release of the product, internal testing sessions, public beta tests, or any other key points in time where your team needs to focus their efforts on testing.

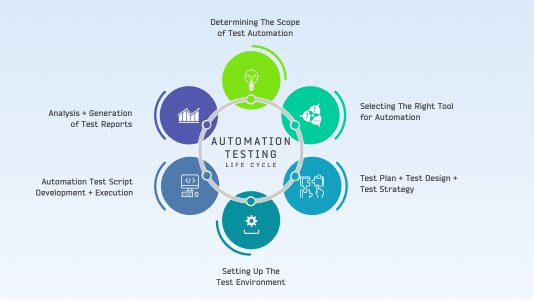
### 13) Test deliverables

Your testing document should include a list of all the deliverables required for testing. These should be linked to the steps in your schedule so everyone knows exactly when they need to be ready for action.

### 14) Test automation

If your software is particularly complex and requires a vast number of test cases, you may want to consider [software test automation](https://www.globalapptesting.com/blog/everything-there-is-to-know-about-automated-testing).

Automating the process means testers can accomplish more in less time, which boosts productivity and significantly reduces the overall cost of testing. You might even be able to utilize a [mobile bot](https://www.automationanywhere.com/company/blog/product-insights/bots-in-your-pocket-get-the-world-s-only-mobile-app-for-bots) to speed up testing activities.



[Image source](https://www.lambdatest.com/blog/all-you-need-to-know-about-automation-testing-life-cycle/)

## Conclusion

A solid test plan is an important part of [dev project briefing](https://www.wearedevelopers.com/magazine/the-importance-of-a-good-briefing-for-every-dev-project). Your testing document should be transparent, concise, and flexible, and adapt to changes in your schedule or environment.

This ensures everyone on your team is working toward the same goal and that nothing gets missed along the way.