Marker.io's Website Test Plan Template

💡 HOW TO USE THIS TEMPLATE

1. **Make a copy of this document** (File -> Make a copy), save that copy as [MASTER TEMPLATE];
2. Everytime you've got a new website to test, make another copy and change the name to: [WEBSITE]_[DATE].
3. You can also integrate the template into your favorite project management software.

Test plan steps

In this template, we include:

1. How to map out the scope of website testing and test objectives
2. Testing strategy and methodology
3. Test environments
4. Test deliverables and reports
5. Resources: roles, responsibilities, testing tools
6. Website test plan schedule: Including testing phase milestones
7. Defect reporting & Issue management
8. Test metrics and reporting
9. Website testing approvals & stakeholder sign-off

Let's explore these components.

1) Scope of website testing activities and test objectives

Before website testing can start, the project leader or QA lead needs to note the scope of the test activities and objectives.

Example: The person responsible needs to answer the following questions about the test plan (or variants based on these):

- Is this a completely new website?
- Or is this a website redesign/overhaul?

- What are the aims and goals of this website (e.g., eCommerce, SaaS, informational)?
- What features, functionality, and tasks (e.g., user-based Jobs to be Done) need testing?
- Does this website need testing in different environments, and if so, how extensive should environment (e.g., OS, device, browser, etc.), alpha, beta, or live user testing be?
- What is “done”, a successful outcome from this website testing project?

How-to: It makes sense to have a template you can use for every testing phase. That way, once a website is ready for testing you’ve already got a standard process to run through.

If a website is larger, more complex, or there’s a bigger budget for more involved testing, then the scope can be adjusted accordingly. For example, a startup SaaS might only have a small 10-page website. It won’t involve much testing, so a simple plan can be used to implement this.

Compared to an eCommerce with 1000 pages and products and dozens of features. A site like that will involve a lot more testing!

Every website will involve a slightly different approach. However, if you start with a basic template/framework then the process of website testing will be easier.

2) Testing strategy and methodology

Website testing should be simple. However, that depends on the size and scale of the website and what needs testing: functionality, usability, performance, or security.

Example: There are a number of approaches you can take to website testing. Either this project will need only one or two, or you might need all four. Here are the choices:

- **Functional:** To make sure a website functions as expected.
- **Usability:** Testing a website from a user’s perspective, for the user experience (UX). This testing can involve alpha, beta, or other live environment human testers.

- **Performance:** Testing a website for everything from loading speed to how it functions in different environments (e.g., OS, device, browser, etc).
- **Security:** Testing for security purposes; may involve working with a cybersecurity expert to ensure a website is as secure as possible, especially if it's going to take payments or secure customer data.

How-to: Based on the information from the first phase of testing (mapping out the scope and objectives), decide which approach or approaches make the most sense for this website.

A QA lead might prefer the Agile methodology if website testing needs to be implemented in a series of sprints. Depending on the size of the website, project scope, time, and resources dedicated to testing, your team may need to implement every type of test.

3) Test environment and test approach the environment requires

Setting up the right test environment is crucial to the success of website testing.

It's also important to make sure any test website isn't accessible to search engines (otherwise that version of the site could get indexed before the finished one), or for anyone to find it either by mistake, or cyberattack.

Example: In most cases, a staging site that's password-protected and not indexed is the best option for a test environment.

This should be an isolated environment that mirrors the production environment, e.g., "staging.yourwebsite.com."

💡 **Pro Tip:** It should be a clone of the production site with a separate database, to prevent data loss or corruption.

How-to: Make sure this staging site is:

- Not available to the general public
- Configured with a **.htaccess** file to prevent search engines from indexing the site
- Password-protected on every page.

4) Deliverables: test data and test reports

Test data and test reports are the basis of the actions your team takes to make changes, improvements, and fix bugs.

Example: When testing a website, you need to collect the following deliverables (as a minimum):

- **Test reports from the test cases** you've run. E.g., we asked 20 people to buy a variety of items on the site. Result: 15 managed, 5 failed–here are the reasons why, and this shows we need to do X, Y, Z, etc.
- **Website feedback**: Design and user experience (UX) feedback is always important at this stage to ensure the site has achieved the project scope and goals and will perform well with the target audience.
- **Bug and defect reports**: Collecting bug reports is a normal part of the QA and testing process. The more you collect at this stage, the quicker they can be actioned, and the fewer issues the website should have once it goes live.

How-to: Before starting testing, know what reports it will produce and how the action items will be implemented.

Ideally, make sure that feedback and bug reports go straight into your PM tool so they can be actioned.

💡 **Pro Tip:** Even when the above is automated, manually check the bug reports and delete duplicates to avoid doubling up your team's workloads.

5) Resources: roles, responsibilities, testing tools

When a website goes to QA testing – either internally, with a client, or with external testers – a **QA lead**, to ensure the end-to-end QA, UAT, and testing process is managed effectively.

Example: You can assign roles, responsibilities, and testing tools based on the scope of the testing and objectives, and the **test plan *(outlined below).***

Once feedback and bug reports come in and the testing process closes, the QA lead should assign tasks to devs, engineers, UX/UI designers, or other members of the team to make any changes the website needs.

How-to: Make sure the team knows who's responsible for managing testing, who's actually implementing it, and who's coordinating with the client or internal stakeholders about the site (this could be the same person in a small team).

💡 **Pro Tip:** For everyone testing your new website, we highly recommend using a dedicated bug tracker and website testing tool.

6) Website test plan schedule: Including testing phase milestones

Next, you need to decide on the test plan schedule and project milestones.

Example: Milestones in testing could be as simple as: A) run test > B) test report > c) implement bug fixes.

Or they could be more complex and multi-layered, especially if testing involved dozens or hundreds of beta testers and automated testing to see how a website performs across different browsers, environments, and operating systems.

How-to: Mapping out the website test plan depends on the following:

- The types of tests involved; e.g., functional, UX, bugs, defects, security, etc.
- How many people are involved in website testing?;
- Following on from that: the scope of the testing. It could be a small group or it could involve 100s or 1000s of beta testers worldwide;
- Whether the tests can be done concurrently or they need to be done based on any dependencies between the tests;
- Whether the testing is manual, automated, or semi-automated.

7) Defect reporting & Issue management

Defect report, also known as bug reporting and issue management is a crucial part of website testing. Ideally, your team should perform internal QA before giving a client or beta testers access, to ensure the most obvious bugs have already been identified and fixed.

Example: Every defect, bug, or issue tracking report needs to include the following:

- **Title**: summarize the nature of the issue.
- **Summary/issue description**: provide more detail. As the reporter, including what you expected to happen vs. what actually happened.
- **Status**: not started, in progress, resolved, etc.
- **Priority**: indicate how critical the issue is.
- **Issue type**: bug, feature request, enhancement... (feel free to customize this!)
- **System**: operating system (OS) used at the time.
- **Browser**: what browser was the reporter using when the issue/bug happened?
- **Metadata/environment info**: extra information on the environment (e.g., screen size, zoom level, and other info).
- **URL**: on what page of the website or app did the issue occur?
- **Screenshot**: crucial so QA teams/engineers can see what the problem is. if you can, add a session replay and/or video recording.
- **Console logs**: logs of any JavaScript error accessible through developer tools.
- **Reporter**: who's reporting the issue?
- **Date**: date of the report or due date.

How-to: You can collect these reports in several different ways.

💡 **Pro Tip:** Use Marker.io to collect live website feedback from testers, team members, clients, and users before, during (and after) a new website goes live.

8) Test metrics and reporting

With website testing, the plan is the outline of the approach. Testing is the throughput, and reports or relevant metrics are the outputs.

Make your outputs easy to understand and implement, especially when it comes to bug reporting and issue management (see above).

Example: The reports and metrics from testing are dependent on the inputs and throughputs from the process.

All of this depends on the size, scale, and complexity of the website and testing process. For smaller websites, the testing reports and metrics shouldn't be too complex or take too long. With larger sites, such as an

eCommerce or marketplace, testing could produce dozens of reports and useful metrics.

How-to: Here's a checklist of the most common test metrics and reports that website testing should produce:

- **Test case reporting:** Includes how to run a test, the entry and exit criteria, inputs, outputs, and pass/fail criteria.
- **Number of test cases that pass:** A higher percentage that pass is a positive metric for any new website. If too many are failing then something is seriously wrong.
- Bug defect tracking: Often measured as the **Defect Detection Efficiency (DDE)**, the total number of bugs identified during testing.
- **Time to bug resolution:** The time it takes to resolve all of the bugs detected.

9) Website testing approvals & stakeholder sign-off

The final stage is getting approval from the client or internal stakeholder that testing objectives have been met and the website is ready to go live.

Example: In most cases, it makes sense to have a checklist at this stage.

Refer back to your PM suite to ensure all tasks have been completed to your satisfaction. Test anything again that needs testing to ensure everything is working as it should.

How-to: Getting sign-off once you've gone through every stage of website testing should be the easy part. It's the last step on the road to getting a new website designed, developed, tested, and launched.

Run through a checklist of the following with your client or internal stakeholders:

- Features and functionality compared to the scope and project goals;
- A list of all bugs, errors, and defects that have been fixed checked against the ones reported during testing;
- A list of all functional, design-based (UX/UI), web copy, or image changes made against the visual and other feedback submitted during testing.

Once all of that's been agreed then the website can be signed-off and pushed from staging into being a live site.