# Load Testing 101:

## Essential Tips for Testers & Developers

LoadUIWeb Pro

by SMARTBEAR

# Load Testing 101: Essential Tips for Testers and Developers

## Contents

## Introduction

Over the last decade, the importance of load testing has skyrocketed. What was once a simple pre-deployment exercise to ensure a web application could handle the load of multiple users has become an intrinsic part of software development and improvement.

At the same time, testers themselves are gaining recognition. No longer a secondary skill within computer science studies, software testing is becoming a specialized career on par with developers themselves. Testing teams and individual testers are now critical players that need to be valued and better understood by anyone involved in software.

We can trace this move to three main trends: the rising popularity of Agile approaches, the interest in collaborative coding as a means of software innovation, and the increasing use of software in all critical domains like healthcare and security. As a result, load testing needs to be done more frequently, more effectively, and more efficiently.

It's also created a need to simply train more people in basics of load testing. Given that our lives increasingly rest on software functioning properly—whether it's in medical devices, transportation, communications, defense, or entertainment—we really need it to work. After all, who enjoys using software that doesn't do what it's supposed to do? No one! So, that's what this eBook is ultimately about—a sort of "Load Testing 101" manual for new and aspiring testers.

This eBook is mainly based on the functions of our load testing tools, LoadUIWeb (free) and LoadUIWeb Pro, but it also serves as a general introduction to load testing. It's broken down into four basic steps: Prepare, Record, Test, Analyze. In the first chapter, you'll learn how to prepare your web application for load testing. The second chapter lays out how to record scenarios, and the third section covers testing that matches real-life circumstance. Finally, we cover the fundamentals of understanding all the data collected with tests, and the best way to use it. Enjoy!

You've worked hard developing your Web application. Maybe it's nothing pioneering, but is one of the millions of unsung software heroes that make our lives function everyday. Then again, maybe it will change the world.

Either way, you know you need to test it - and not just once. Testing should be performed at each step in the development cycle and should continue after the application is live. While it can be frustrating that a tester's job is never done, it's important to take consolation that with each testing and remediation cycle, the application improves.

Applications go down under load for two reasons: Either developers didn't load test or, worse, they took the time to load test but didn't prepare properly. Without adequate prep work, a load test can't find all the issues that it should.

So, how do you best prepare for a load test?

## 1. What do you really need to know?

Determine what you want to learn about your application or system. Each type of test is run differently, and looks at your application in a different light. So, you'll need to run different types of tests based on what you hope to find out. For example:

If you hope to discover how your application performs with little or no load in order to get a baseline, you will run a single user test.

- If you hope to determine how your system will perform under normal expected load, you will run a load test.
- If you hope to determine the breaking point, the point where your application either stops responding or responds so slowly that it is unusable, you need to run a stress test.
- If you want to know if your application has memory leaks, you will want to run an endurance or soak test.

## 2. Decide on a number of users

If you are going to load test, how many virtual users do you want to simulate? In order to answer this, you will want to approximate how many concurrent users may visit your site, and that depends on the time of day. Many testers just take a guess. Instead, talk to your architect, talk to your marketing people, and look at the performance specifications. If you want to know some concrete statistics from a historical perspective, go directly to your analytics reports. You may even want to ask your engineers how many concurrent users they designed the application for, and your product owner/marketing base their projected numbers based on promotional activities. Plan to test that number and some percentage above it.

**Note:** If you must run a post-production load test for some reason, be sure to schedule your tests for a time when actual users can be minimized or eliminated.

## 3. Study your analytics

Do not pretend to know how your customers use your application. The only way to truly understand your users is to study history (i.e. analytics). By studying your analytics, you will be able to create tests that are actually representative of your actual users – as opposed to tests that you think are representative of your users. In this regard, analytics are a tester's best friend!

## 4. Gather your team

You need to involve a number of people in the testing effort: Developer, Network Engineer, DBA, and Business Owner – to name a few. All of these individuals have a vested interest in making the application successful, and each will approach the problem from a different angle. The correct solution will not fall directly into one of these buckets, but will be a combination of two or more. Make sure each is available during testing to:

- Monitor their area of expertise.
- Provide balanced feedback.
- Gain a sense of ownership for the health and performance of the application.

### 5. Prepare your browsers

Use testing software that brings you as close to your actual users' experience as possible. You should be able to record your scenario in the browsers of your choice, but you also need to anticipate the browsers your users will most likely use. Consider the countries and regions where you anticipate high usage, and research the most used browsers. You'll need to have these installed on your machine to begin testing. Then you need to make sure your load testing software emulates as closely as possible actual user behavior. This includes:

- Parallel thread processing.
- Think time
- Multiple Concurrent Scenarios
- Complex Scenarios
- Parameterization
- Generating load from multiple agents (network/cloud)

### 6. Be prepared to test your production application

While it is valuable to test your application when it is in a staging environment, for a number of reasons this can leave some holes in your testing.

- Staging environments are not often exact duplicates of production.
- Staging environments are often accessible from only inside the firewall.
- There is something to be said for testing the same system that you are gathering information about.

## 7. Set aside time to analyze results

You should be prepared to spend some time analyzing test results as a group (remember all of those people that were present during testing?). Results need to be looked at carefully to ensure bottlenecks/errors/weaknesses are really understood and that the remediation is effective. Make sure to reach out to everyone and schedule adequate time.

Results need to be looked at carefully to ensure bottlenecks/errors/weaknesses are really understood and that the remediation is effective.

## 8. Set aside time to make changes

Also, allow time in your schedule to actually implement the changes that you determine are needed! Different remediations will have differing costs in terms of time. Remediations such as implementing a caching strategy, refactoring code, database optimization and hardware upgrades have a wide range of costs to implement in terms of both time and money. As an example, adding more hardware will require time to order the hardware, receive the shipment, test the new hardware, install software and data, test, install into the network, and test some more. This can be weeks or months! Unless, of course, you are in the cloud. In which case, it takes less than a day. Many companies are opting to move to cloud infrastructures, offering an unlimited number of environments without the need for additional hardware costs and time constraints. However, it's always advisable to load test within your firewall as well. Some load testing tools, like LoadUIWeb Pro, allow you to do both. Our free tool, LoadUIWeb only tests from a single machine up to 400 virtual users, which is adequate for many applications and a great place to start learning load testing.

## 9. Plan an Agile testing methodology

Once you remediate, it is time to test again! The saying, "testing is a process, not a destination" is very true. Each time a bottleneck is uncovered and corrected, another one rises to take its place. It is important

to plan an Agile testing methodology, whereby performance testing is baked into each step of the development cycle. Additional testing should be performed:

- When code is modified or updated.
- When environment/infrastructure changes are introduced.
- When changes are made to the application server or DB server.
- When traffic spikes are anticipated.

**Take a deep breath and relax!** You've already done most of the heavy lifting. Now that you've taken the time to really prepare, load testing your application will help you continually improve your product and your business.

## How Role-Playing Games Can Save Your App

It's easy to get swept up in role-playing video games. Who can resist the temptation to be anyone you want in a fictional world filled with unending excitement? The problem is, you inevitably end up wondering whether you're involved in a cutting edge virtual reality or something more akin to those speaking Old English at a Renaissance fair. You never really know, so it's probably best not to brag too much at work about your adventures the night before.

Creating scenarios in order to perform load tests is a major aspect of both deploying and improving web applications.

However, there is one area where role-playing games are certain to win you points in your career. Creating scenarios in order to perform load tests is a major aspect of both deploying and improving web applications.

### How to record scenarios

The first thing to do is to determine the roles you will define for use in your test.  A role is equivalent to a certain type of user that will visit the tested website, and the steps they will take while visiting.

If the tested site is a retail site, for example, you might have the following roles:

- Browse and leave
- Browse, add something to the cart and check out

If your tested site is a restaurant site, your roles might look something like this.

- Browse menu and find directions
- Look at hours of operation and make a reservation

It's best to choose at least three of the most common pathways through your site, and add a few uncommon routes as well.

Next, you need to break these roles down by percentage of traffic. A typical retail site may have 95% of users browsing and leaving, and 5% (or less!) actually making a purchase.

| PCT | Role |
|---|---|
| **95** | Browse and Leave |
| **05** | Browse, add something to the cart and checkout |

The combination of these two roles, or scenarios, will represent actual site traffic.

After each scenario is recorded, you need to verify it individually. This involves running a single virtual user for a single pass through the scenario. This step should never be forgotten.

Now you're ready to start testing. By recording scenarios that imitate actual user traffic, you're setting the stage to greatly improve customer experience and, if you're into ecommerce, get that percentage of purchases above 5%!

Whether it's an elementary school math quiz, a college history exam, or a software development team's load test, we always want our tests to emulate real-life conditions. Otherwise, what's the point of testing?

In this chapter, we'll discuss ways you can ensure your load tests match reality.

Some of the best perks about our free load testing tool, LoadUIWeb, are the settings that aid the process of generating a realistic load test. Of course, like all load testing tools, you can specify the number of virtual users to be simulated, but you can also set certain conditions that easily create more powerful and reliable load tests. No matter which load testing tool you choose, make sure it allows you to set some version of these basic conditions.

## Ramping up virtual users

When running performance tests, it's not desirable to start all virtual users at the same time. Starting all virtual users at the same time can create artificial bottlenecks in certain parts of the application – such as the login process. In LoadUIWeb, you can elect to set a load profile whereby you start with a small number of virtual users, and increase that number over time. For example: you start with a single virtual user and add one virtual user every two seconds until you reach a certain number of simultaneous users, and then hold that number for the duration of the test.

## Setting load duration

In order for you to run a test with large numbers of virtual users, you will need to set duration for your test. By setting duration, each virtual user will execute its scenario and when it reaches the end it will start over – thereby maintaining the level of load. For example, if your scenario takes two minutes to be executed, but you run a test for 10 minutes, the

scenario might be executed five times by each virtual user. This function can be found under the “continuous load” tab of the load profile in LoadUIWeb. Merely check the enable box, and set a time.

## Parameterizing tests

While recording a scenario, you may need to specify some parameters that will be used for further test runs. For example, you can enter some search terms, user names and so on in the application’s fields. However, it is not a good practice to play back a test with the same recorded data for each user as it does not simulate the real-life conditions. To solve the problem, LoadUIWeb allows you to parameterize your load test using special variables. A variable can store desired data and your requests can use this data during test runs.

## Emulating browsers, network connection bandwidth and browsing speed

Real users visit a website using different browsers, the bandwidth of user’s Internet connections can vary significantly, and they spend different amounts of time on each page. You can configure all these parameters when creating load tests in LoadUIWeb. This means you can specify different browsers and connection bandwidth for different groups of virtual users. You can also specify think time for each tested page (we call this think time, as it simulates the time when a user is viewing the page and thinking). You can easily randomize the think time for simulated scenarios to better emulate real user activity.

## Simulating concurrent requests

To download a Web page, modern Web browsers send requests to the server using several simultaneous connections. These parallel requests download images, scripts, CSS files and other resources located on the page. Unlike many other load testing tools, LoadUIWeb automatically detects and simulates parallel requests - that is, it sends requests

exactly like your browser did when you recorded your scenario. This makes the simulation closer to real-world conditions.

Most load testing tools collect exhaustive information for each load test run. Traditionally (if we can say that about load testing), you needed to have pretty extensive knowledge in order to interpret the majority of load testing data in any kind of meaningful way. This, unfortunately, has resulted in countless load tests that generated tons of useless data.

## Analyzing Data in Load Testing: What You Really Need to Know

More often than not, developers and QA managers come away from a trial of load testing software with little more than the number of users that will crash their system. Unless they have a professional load tester on staff, most development teams don't have the time, resources, or knowledge to garner all they could from their load tests.

> If you know what to look for, improvements in graphics and user interfaces like those in LoadUIWeb have made interpreting data much easier.

To think of all that data just sitting there, unable to fulfill its life purpose of Web application betterment is sad. Luckily, improvements in graphics and user interfaces like those in LoadUIWeb have made interpreting data much easier—if you know what to look for.

Here's a list of the most important results in load testing and how you should be working with them.

### Page load time

You absolutely want to know the average page load time for each page in your scenario. You might have a strict Service Level Agreement (SLA) that mandates how quickly pages must load, or may just want to know what this number is. It is also important to know if one page takes longer than others to load—this indicates a bottleneck in your application.

### Response load time

Just knowing page load time is not enough. If a page is slow, you need to know why. Being able to look at average response times for each response really gives you a detailed look into where the time is being spent.

### Errors and warnings

You need to know which errors and warnings were generated and at what level of load. This is especially important information to see in chart format. It is important to see which errors and warnings are generated and be able to see how that changes as load increases. A common error at high levels of load is “Server Error 500’s.”

### Request and response throughput

It’s important to see the amount of data going to and coming from the tested system. This is especially important in a case where load is increasing, but bandwidth reaches and maintains a plateau. In this case, it becomes apparent that bandwidth is being throttled at some point in the process, possibly at the firewall, and often this can be fixed by changing a setting.

### Hosts

Because so many of today’s websites call out to a plethora of additional hosts for things like content delivery networks, ad servers, analytics servers, social media and syndicated content, it is important for these sites to be enumerated in your reports. It’s equally important to be able to view all of the calls to a particular host. If a host is called from your pages, the response time for those requests will add to the time it takes your pages to render. You must be aware of and possibly take action in the case where a certain call takes a long period of time to respond.

## Conclusion

We hope you've found this eBook informative, inspiring, and a resource for future reference. If you're interested in our free load testing tool, click here to find out more, or go ahead and download LoadUIWeb. If you find you're in need of more functionality, customizability, need to test from the cloud or more than 400 virtual users, LoadUIWeb Pro will be a better choice.



### About SmartBear Software

More than one million developers, testers and operations professionals use SmartBear tools to ensure the quality and performance of their APIs, desktop, mobile, Web and cloud-based applications. SmartBear products are easy to use and deploy, are affordable and available for trial at the website. Learn more about the company's award-winning tools or join the active user community at http://www.smartbear.com, on Facebook or follow us on Twitter @smartbear and Google+.

SMARTBEAR

SmartBear Software, Inc. 100 Cummings Center, Suite 234N Beverly, MA 01915
+1 978.236.7900 www.smartbear.com ©2013 by SmartBear Software, Inc. Specifications subject to change.
EBOK_LUIWP_20130729_V1_LOAD-TESTING-101