**Why Performance Testing is a Good Idea**

Software with bugs and glitches that impede or limit its use can spell the difference between business success and failure, the ability to attract/retain users or lose them to the competition.

The near ubiquitous use of Agile development has fundamentally changed how teams performance test their software before releasing it for consumption.

**How Has the Performance Testing Process Changed?**

In the past era of Waterfall development, there was a period of time reserved to execute the load, stress and [performance](https://mentormate.com/guide/unlocking-software-testing/) testing process. The testing had been designed and planned for months. During the designated window, a long list of tests would be created and executed.

Testing tools like LoadRunner were commonly used. They ware powerful, but they were also complex to set up. As Waterfall gave way to the short release cycles of [Agile](https://mentormate.com/guide/agile-software-development-remote-teams/), it became clear that the old “plan way in advance and then wait for your testing window” performance testing approach just did not fit. The bottleneck created waiting for developers to finish work and hand it off to testers was in fundamental conflict with [Agile](https://mentormate.com/blog/agile-software-development-plan-offshoring/).

The testing needs of today are fundamentally different. A performance test strategy is needed even at the outset of the project. For solutions developed using Agile methodology, working software is delivered at the end of each two-week sprint. Testing teams can no longer rely on a dedicated window for testing. They must test iteratively as the solution is being built.

To test effectively, validate development assumptions and keep pace with engineers, software quality teams must consult with both the client and project teams to understand the primary business drivers and implement testing to validate the functionality and its performance.

**Tools Used in the Performance Testing Process**

The same [tools](https://mentormate.com/blog/complete-guide-automated-software-testing-tools-comparison/) are used for performance, load and stress testing. The top ten most popular performance, load and stress testing tools are:

[**WebLOAD**](http://www.radview.com/webload-download/) — WebLOAD’s IDE offers a variety of features including response validation, correlation, messaging, parameterization and debugging. It can generate high virtual user load in the cloud or locally.

[**LoadUI NG Pro**](https://smartbear.com/product/ready-api/loadui/overview/) — LoadUI NG Pro stands out because it allows teams to repurpose functional API tests built using SoapUI Pro.

[**Apica LoadTest**](https://www.apicasystem.com/load-testing/) — Apica LoadTest allows teams to access on demand and continuous integration load testing.

[**LoadView**](https://www.loadview-testing.com/) — LoadView performance testing is 100% managed in the cloud. This tool offers an affordable option to control infrastructure testing expense.

[**Load Xen**](https://testcollab.com/loadxen/) — The ability to simulate a large number of parallel users, flexible scripting and request data randomization are a few of the reasons teams choose Load Xen.

[**Apache JMeter**](https://jmeter.apache.org/) — Apache JMeter offers the ability to performance and load test the following: web solutions, SOAP/REST webservices, FTP, database via JDBC, LDAP, message-oriented middleware and more.

[**LoadRunner**](https://saas.hpe.com/en-us/software/loadrunner) — LoadRunner just rolled out a variety of new features from JSON manipulation APIs to Git integration in VuGen.

[**Appvance**](https://www.appvance.com/)— Appvance offers a unified test automation platform.

[**NeoLoad**](https://www.neotys.com/neoload/overview) — Descriped as a full performance testing tool, NeoLoad offers supprt for WebSocket, HTML 5, HTTP/2, GWT and others.

[**LoadComplete**](https://smartbear.com/product/loadcomplete/overview/) — LoadComplete offers teams the opportunity to scalability, stress and load test website an›d web applications.

At MentorMate, we use Apache JMeter (for web and API testing) and SoapUI (for API testing). Why do we choose them? These options are new-generation, open source tools that are constantly being evolved and improved by developers all over the world.

**What Is the Cost of Poor Performance?**

Performance testing validates the important non-functional characteristics of applications, such as:

* How fast is the system?
* How much load can the system handle?
* Under what conditions will the system fail?

While less commonly taken into consideration than verifying user interactions, performance testing can be a smart strategic solution for businesses who want to create an overall positive user experience for their customers.

This testing provides valuable [performance](https://mentormate.com/guide/interactive-software-test-planning-tool/) intelligence to help businesses assess whether the solution answers the needs of the business now and whether that will continue into the future.

The financial cost of poor performance and service interruptions varies widely according to the type of application affected.

* Customer flight
* Partner distrust
* Damage to the corporate image
* Reduced employee productivity

Completing a round of performance testing can be the differentiator between two businesses, especially during periods of extreme expected traffic.

Consider this scenario, a retailer added gamification to its site. This increases load to existing servers during peak shopping times like Black Friday or Cyber Monday. One result of performance testing, in this case, could be delivering the recommendation use a separate server to host the gamification to avoid slowdowns or crashes during peaks for high load.

There are three common types of performance testing: performance, load and stress testing.

**What Is Performance Testing, and What Are the Types?**

For teams asking, “What is performance testing,” this type of testing helps businesses assess how components of a system perform in particular situations. Resource usage, scalability and reliability of the product are also validated during this testing. Performance testing is a subset of performance engineering, focused on addressing performance issues in the design and architecture of a software product. A performance test is a load test limited by the load defined by the specification of the application. The aim is to verify or confirm that the application will work at the planned performance.

**How Fast Is the System?**

The purpose of the performance testing process is to answer the question: How fast is the system? The primary goals for performance tests include:

* Number of concurrent users required for the system
* Response time of the system for each/all transactions or a particular process/workflow
* Acceptable limits of system utilization CPU, memory
* Validating that the application performs properly
* Validating that the application conforms to the performance needs of the business
* Finding, analyzing and helping fix performance problems
* Validating the hardware for the application is adequate
* Doing capacity planning for future demand of the application

**A Common Performance Test Pain Point**

The biggest challenge to these types of testing is test environment. Ideally, tests should be conducted on real scenarios or the production environment. If a replica of the production environment cannot be provided, performance testers problem solves around the limited hardware resource availability in order to get the accurate results.

**Example Scenario**

If we want to check the email functionality of an application, it could be flooded with 1,000 users at a time. Now, 1,000 users can fire the email transactions (read, send, delete, forward, reply) in many different ways. If we take one transaction per user per hour, then there would be 1,000 transactions per hour. By simulating 1 transaction/user, we could execute performance test of the email server by occupying it with 1,000 transactions/hour. In completing this test, we would monitor load time, response time and latency.

**Load Testing**

[Load testing](https://mentormate.com/blog/software-testing-best-practices-changing-development/) is performed to determine a system’s behavior under both normal and at peak conditions. We can assume that businesses want their applications to constantly perform. Though, we concede there will be certain times when the application will have a heavier load to bear than others.

Load testing helps to identify the maximum operating capacity of an application as well as any bottlenecks. This helps businesses determine the elements causing the degradation, if any. For example, if the number of users increase, how much CPU/memory will be consumed and, what is the network and bandwidth response time?

**How Much Load Can the System Handle?**

The primary goals of load testing are:

* Find metrics for system performance under high load.
* Define the maximum amount of work a system can handle without significant performance degradation.

**Example Scenario**

Black Friday is the worst possible time for an e-commerce website to crash. For high profile companies, even a few minutes of downtime can become a financial and public relations nightmare.

**Stress Testing**

Some systems require more specific treatments. Stress testing studies the way that the application works under unexpected conditions, such as how a system keeps working when subjected to extreme workloads or when some of its hardware or software has been compromised.

Stress testing also provides insight into how the app crashes and the time needed to repair the solution after a crash. The other important aim is to ensure that the app does not throw exceptions containing sensitive information during the crash.

**Under What Conditions Will the System Fail?**

The goal of the stress testing is to analyze post-crash reports to define the behavior of application after failure. The biggest issue is to ensure that the system does not compromise the security of sensitive data after the failure. In a successful stress test, the system will revert to normality along with all its components, after even the most terrible breakdown.

**Example Scenario**

If an app where users store images or data crashes and during the failure user data is displayed, this could compromise user privacy or safety. Consider the outcomes if a behavioral health app crashed and displayed user data. At worst, that information could be viewed by a user’s employer resulting in a breach of confidentiality or a less secure employment.

**Improving Software Quality**

Testing for performance issues can significantly improve the overall quality of a software product from the start of the development effort. Planning and executing successful performance strategy can be daunting task.