**Endurance Testing:**

* It is a type of non-functional testing.
* It is also known as Soak testing.
* Endurance testing involves testing a system with a significant load extended over a significant period of time, to discover how the system behaves under sustained use. For example, in software testing, a system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly.
* The goal is to discover how the system behaves under sustained use. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good or better than at the beginning of the test.
* It is basically used to check the memory leaks.

This type of testing is performed at the last stage of the performance run cycle.

Endurance testing is a long process and sometimes lasts for even up to a year.

This may include applying external loads such as Internet traffic or user actions. This makes endurance testing differ from [Load Testing](https://www.guru99.com/load-testing-tutorial.html), which usually ends in a couple of hours or so.

*Endurance means capacity so in other words, you can term Endurance Testing as Capacity Testing.*

In Endurance Testing following things are tested.

* **Test memory leakage**- Checks are done to verify if there is any memory leakage in the application, which can cause crashing of the system or O.S.
* **Test connection closure between the layer of the system** – If the connection between the layers of the system is not closed successfully, it may stall some or all modules of the system.
* **Test database connection close successfully**- If the database connection is not closed successfully, may result in system crash
* **Test response time** – System is tested for the response time of the system as the application becomes less efficient as a result of the prolonged use of the system.

***Endurance Testing Example:***

While **Stress testing** takes the tested system to its limits, **Endurance testing**takes the application to its limit **over time**.

For Example, the most complex issues – memory leaks, database server utilization, and unresponsive system – happen when software runs for an extended period of time. If you skip the endurance tests, your chances of detecting such defects prior to deployment are quite low.

**Endurance Testing Tools**

* WebLOAD
* LoadComplete
* [Apache JMeter](https://www.guru99.com/jmeter-tutorials.html)
* [LoadRunner](https://www.guru99.com/loadrunner-v12-tutorials.html)
* Appvance
* LoadUI

## Advantages of Endurance Testing

* It helps in determining how workload can the System Under Load handle.
* Provides accurate data that customer can use to validate or enhance their infrastructure needs.
* Identifies performance problems that may occur after a system has been running at a high level for longer period of time
* Typical issues are identified in smaller targeted performance tests, which means it ensures application remain available even when there is huge load in a very short span time.
* The endurance test is also used to check whether there is any performance degradation after a long period of execution

## Disadvantages of Endurance Testing

* It is often hard to define how much stress is worth applying.
* Endurance Testing could cause application and/or network failures that may result in significant disruption if[Test Environment](https://www.guru99.com/test-environment-software-testing.html)are not isolated.
* Permanent Data loss or corruption can occur by over-stressing the system.
* Resource utilization remains very high after the stress is removed.
* Some application components fail to respond.
* Unhandled exceptions are observed by the end user.

# Scalability Testing

* **It** is a non functional testing.
* It measures performance of a network or system, when the number of user requests are scaled up or down.
* The purpose of Scalability testing is to ensure that an application can handle the projected increase in user traffic, data volume, transaction counts frequency, etc.
* Scalability testing lets you determine how your application scales with increasing workload.
* Determine the user limit for the Web application.

**Portability Testing**

Portability testing refers to the testing with ease of moving one product or software from one environment to another.

One example : includes testing of a product which is well functioning in Windows 7 and measuring its behaviour in Windows 8. This is measured in terms of the effort involved in the task.

The main use cases followed in portability testing is when the application is designed in such a way that it is bound to be moved from one hardware platform, operating system or web browser to another.

Following are some of the real life use cases:  
  
• Designing software to run on Windows operating system and Mac OS X.  
  
• Developing an application to run on Google’s Android, Apple iOS, and Windows’s operating system.  
  
• Developing an application that is compatible on web browsers viz. Internet Explorer X,Mozilla Firefox and Google Chrome.

## Baseline Testing

* Baseline testing refers to the validation of the documents and specifications on which test cases are designed.
* Baseline, in general, refers to a benchmark that forms the base of any new creation.
* In software testing, this refers to benchmarking the performance of the application.

This test forms the base for other testing to compare the performance of a new application or unknown application with a known standard of reference.

For example, if a particular application is known to give a good performance for at least 1000 users at a particular time, then the baseline can be 1000 users. So, a new application should work perfectly for a minimum of 1000 users.  
  
We can think of a software update. With each update, it should have old features plus new ones. Clearing the baseline test ensures that the main characteristics and benchmarks of the application are working as expected.

**Load Testing**

It’s a type of performance testing that simulates real-world load on any software, application, or website.

It examines how the system behaves during normal and high loads and determines if a system, piece of software, or computing device can handle high loads given a high demand of end users.

This tool is typically applied when a software development project nears completion.

Load Testing ensures that your application can perform as expected in production.

Just because your application will pass a functional test, this does not mean that it can perform the same under a load.

Load testing identifies where and when your application breaks, so you can fix the issue before shipping to production.

**SMOKE TESTING V/S SANITY TESTING**

|  |  |
| --- | --- |
| **Smoke Testing** | **Sanity Testing** |
| Smoke testing ascertains that core functionalities of the program are working fine absolutely. | Sanity Testing is done to check either new functionalities or bugs have been fixed properly without going deeper. |
| The objective of smoke testing is to verify the stability of the system to process regression testing in the future. | The objective of sanity testing is to verify the rationality of the system to proceed with more regression testing in the future. |
| Smoke testing is performed either by developers or testers. | Sanity testing is performed by testers only. |
| Smoke Testing is usually documented and scripted. | Sanity testing is not documented or scripted. |
| Smoke testing is a subset of acceptance testing. | Smoke testing is a subset of regression testing. |
| Smoke testing focuses on the entire system from end to end. | Sanity testing focused on selected components of a system. |
| Smoke Testing is like General health check-up. | Sanity testing is like special health check-up. |