**Performance testing life cycle**

As a performance testing company we would like to articulate the life cycle of the performance testing and the associated entry and exit criteria. Like how we have SDLC and STLC models performance testing takes its owns life cycle which can be acronym as PTLC .

Being a load software testing company we want to define and implement the stringent test life cycle phases to ensure we deliver a flaw less application and also a one that satisfies he user in all means.

By the definition, performance testing is all about validating the behavior of application which is hosted on a server under critical load, volume and stress conditions to verify the set benchmark parameters are satisfied before the application goes live.

Phases involved in performance testing life cycle are as follows

**Nonfunctional requirements gathering**

It is essential to understand the performance test requirements as these are intended to serve a completely a different purposes than the functional testing. As a performance tester first we must understand the functionality of the application under test to understand the nature of the application

Understanding the nature of the application is much needed to define the scope and also to identify the business scenarios to be tested.

There have to be through discussions with the business analysts, technical architects and the developers to confirm the nonfunctional requirements in terms of the peak load and the number of iterations needed for a particular scenario that’s identified.

**Entry scale**

The entry base for this phase would be to going through the documentation that is defined to understand the application and its work flows then building questions

**Exit scale**

After a series of fruitful discussions, coming up with all the possible scenarios that are considered to be tested then signing-off the document as that acts as a base line document.

**Performance Test strategy**

In this phase following activities must be considered to be acted upon

Conducting a feasibility analysis in defining the in scope and out of scope requirements

This should be considered as a subsequent action to the NFR that’s defined. Careful study of the requirement then understand what infrastructure is all needed to achieve that then calling out any challenges that would be foreseen.

As a performance tester we must gain control over the functionality as well as the technical internals that are used in application. After understood the requirement then we can come to a conclusion on the below

* If any scenario is not possible within the given infrastructure due to environment that has no downstream connections or no tool support or due to any technical reason we have to bucket that under out of scope with proper agreement with all stakeholders. Also it is keen to understand how user friendly the use case is going to be? Basis that the performance testing company can leverage the infrastructure because quality does matter a lot

Eg: how the transactions take place and what technique is being used to place a transaction and the technologies being used such as kafka, message queue, stremsets, ping, cws authentication.

* The feasibility study also states the SLA for a specific scenario, as we might get into a situation where some R&D is needed to achieve a new task.

**Identification of tools and infrastructure**

Before we jump on to any design or implantation plans we should analyze and choose a best suit tool for the project need. There are two things to be considered before we take a decision

**Tool cost: -** in order to help with cutting the cost for a company selecting an open source would be a good move, if the client or customer has their own preference of opting for a commercial tool we can prefer that one too. The ideology here is selecting a one that satisfies all the needs then selecting a one that can do all for us such as giving good reports, support plugins and should support

**Community support:** - it’s not certainly good move selecting a tool that has very narrow community support, having a good supported community will help us greatly to resolve any problems that we encounter during the usage

Keeping the wider tool support a fixed thing, to comment on the other option of opting a freeware of commercial is always debatable, this can fairly be negotiated with the stakeholders and go for selection

**Popular Performance Testing tools are:**

 Jmeter (open source)

 Load runner (commercial)

 Silk performer (commercial)

 Neo load (Commercial)

**Application performance management tools:**

 App Dynamics

 Wily-introscope

 Perf mon

 N-mon

**Environment and data set up**

As a performance testing company we should always keep a separate environment for NFT testing and it is very much required that the environment should impersonate the production environment. The big reason for this is, NFT testing is going to set up some benchmark or critical parameters for the application as stated below

How many parallel users can access the application?

How scalable the application is during peak load?

How many transactions can be processed when subjected to bottleneck conditions?

The above mentioned parameters can only be determined in an isolated environment, if at all the performance testing is asked to share the environment with system testing team that jeopardizes the productivity of the both the teams.

And in the same fashion it is good have a production data copy loaded to the NFT environment to better replicate customer use cases.

**Assessing the risks and assumptions**

As a performance tester we should be proactively calling out the risks and they should be aforementioned. Below are the some possible risks that can take place. Shouting them out at right times would help the business to make plans to go live

Analyzing the dependency with the development and environmental team to get the support required during application being broke/ downtime instances

 Going live with the presence of a defect that may occur at customer sight

 Can’t test a particular scenario due to the unavailability of the infrastructure in the lower environments

 Testing a feature by taking some deviation

**Performance Test design:-**

As we are done with the discussions on NFT gathering, feasibility study on all scenarios and identification, we should now be in a decent shape to start with deigning the scripts that are to be executed. Most likely we will inject records in DB by hitting an API or traversing the application UI screens and measuring the response times.

**Performance Test execution:-**

Once the scripts are in place and we have the dedicated NFT environment to point to we can start with the execution. The whole execution process consists of following steps

* Run the scripts
* Triage the failures and baseline the scripts
* Monitor the execution to understand any application failures
* Capture the results

**Test result analysis:-**

Once the execution completes as a performance tester we should conduct a careful study on the results. We have to consider comparing the benchmark parameters defined during initial discussions with the results we have captured in performance test tools to conclude whether the AUT is compliant in terms of performance criteria that’s defined. Any SLA breach has to be logged as a defect and we should make the development team look at it.

**Test results circulation:-**

After completion of the test execution for he agreed number of iterations a good results and reports to be shared with all the stakeholders as that pretty much gives the confidence to implement the application or not.

**Documenting the learning’s:-**

As we proceed with tasks every day we learn many things, it is fact that some learning may go unnoticed. But as a performance testing company we should make the team to document the lessons learnt, challenges faced and the associated resolutions as this is going to be a bible for the next iteration. A good retrospection always helps us to gain every which way.

Thanks for the read, hope the blog is informative..keep learning