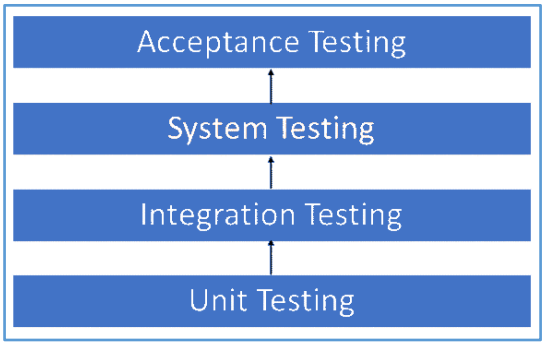
NONFUNCTIONAL TESTING

LEARNING OBJECTIVES

- Know the difference between functional and nonfunctional testing
- Know the different types of nonfunctional tests and the basic implications involved in nonfunctional testing
- Set up and execute test plans in JMeter
- Import and execute test plans in BlazeMeter

NONFUNCTIONAL TESTING

Performed after integration testing and before system testing



Functional / Nonfunctional
Functional requirements describe what the software system should do
Nonfunctional requirements describe how the software system should work

Nonfunctional testing is a type of testing to check nonfunctional aspects of a software application

Example…
Functional requirement: Software system must send emails when orders are placed
Nonfunctional requirement: Software system must send emails within 1 minute after orders are placed

Objectives of nonfunctional testing…

- Produce measurements and metrics for further research and development
- Enhance knowledge about the product behavior and the technologies in use
- Optimize the way the product is installed, set up, executing, managed and monitored
- Increase quality attributes of the product

Quality attributes of a software system include nonfunctional testing parameters such as security, availability, efficiency, integrity, reliability, survivability, usability, flexibility, scalability, reusability, interoperability and portability

DIFFERENT TYPES OF NONFUNCTIONAL TESTING

There are many different types of functional tests, targeted at testing each of different nonfunctional testing parameters
Not all testing types apply to all projects but depend on the nature and scope of the project

Nonfunctional testing types

- Security testing / Maintenance testing / Regression testing
- Reliability testing / Availability testing / Stability testing
- Survivability testing / Recovery testing / Disaster recovery testing
- Performance testing / Stress testing / Efficiency testing / Load testing / Endurance testing
- Volume testing / Scalability testing
- Usability testing / Interoperability testing / Portability testing / Compatibility testing

Security testing
: Focused on weaknesses, threats, vulnerabilities, loopholes…

Reliability testing
: Focused on failure-free operation for a specified period of time…

Survivability testing
: Focused on recovering from failures and reestablishing integrity…

Stability testing
: Focused on ability to function without failures under different stress loads…

Usability testing
: Focused on end users usage of the system…

Scalability testing
: Focused on ability of the system to scale according to needs…

Nonfunctional testing ranges widely from sessions recording end users using the system in a test lab to actual disaster exercises of recovery plans for restoring the system back to operation…

Extensive reports and summaries should be generated for the nonfunctional tests including different types of failures, response times, statistics and similar

MTTF = Mean time to failure	(Difference of time between two consecutive failures)
MTTR = Mean time to repair	(Time required to fix failure)
MTBF = Mean time between failures	MTBF = MTTF + MTTR

Actions can be taken based on nonfunctional testing reports and summaries to improve the system and meet the nonfunctional requirements

Improvement depends upon the characteristics of the system and the problems occurred

Exact numbers are unlikely to be known at the start of the requirement process

TOOLS

Load / Performance
: JMeter / BlazeMeter
: Loadrunner / Neoload / WebLoad / Loadster / WAPT / LoadImpact

Usability
: CrazyEgg / UserZoom / UserTesting / Loop11 / UsabilityHub / Usabilla

Security
: SonarQube / SQLNinja / Wapiti / W3af

JMETER

https://jmeter.apache.org/
Open source software / 100% pure Java application
Designed to load test functional behavior and measure performance
Originally designed for testing Web Applications but has since expanded to other test functions
CLI mode to load test from any Java compatible OS (Linux, Windows, Mac OSX)
Ability to load and performance test many different application / server / protocol types:

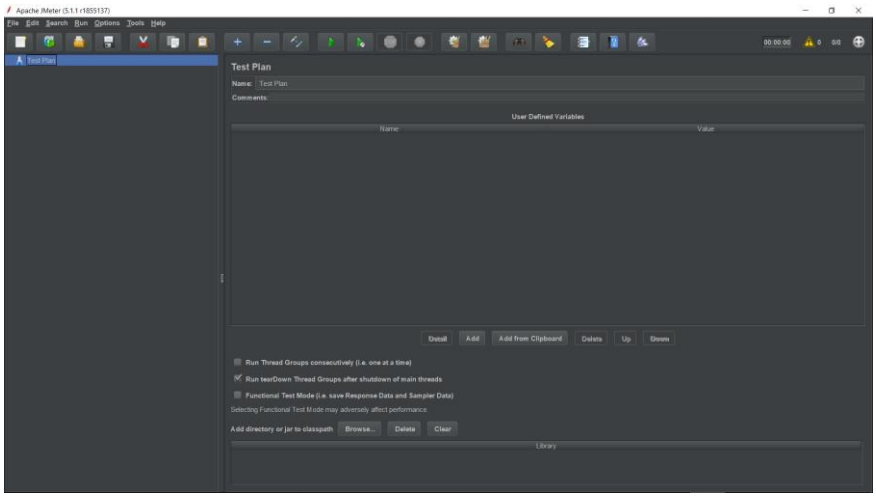
- FTP / TCP
- HTTP / HTTPS
- SOAP / REST Web services
- Database via JDBC
- SMTP(S) / POP3(S) / IMAP(S)
- Native commands or shell scripts

JMeter can simulate browser behavior by sending requests to web or application servers and also parse the responses
JMeter does not execute the Javascript found in HTML pages, nor does it render the HTML pages as a browser does

Download and unpack JMeter binary package
https://jmeter.apache.org/download_jmeter.cgi

Run JMeter jar file
/JMeter/bin/ApacheJMeter.jar

JMeter GUI



Test plan
The Test Plan is a JMeter script and it determines the flow of the test

Create new test plan / Save test plan / Build test plan

Thread groups
Run thread groups consecutively (One at a time)

Set up a thread group

- Number of threads / users
  - Average 200 – 300…
  - Exceptions if too many…
- Ramp up time
  - How long to "ramp-up" to the full number of threads

Example…

NumberOfThreads = 200
RampUpTime = 10
20 threads are started each second

Samplers

HTTP Request

Server / Path / Method / Body / Encoding

Assertions

Response Assertion (Assert HTML and response)

Text response = HTML
Response code / Response message / Response headers
Pattern matching

JSON Assertion (Assert JSON properties in response)

Assert JSON path exists $.properties
Additionally assert value Complete value / Regular expression

Timers

Constant timer Wait for a constant amount of time
Uniform random timer Wait for a random amount of time

Listeners

View results tree

Examine individual requests and responses

View results in table

Sort results

Summary report

Minimum / Maximum / Average / Error

Aggregate report

90% / 95% / 99%

Response Time Graph

Time / Response times

Run -> Clear all

Nodes

Disable / Enable
Toggle

Log

Show / Hide
Errors

Config Elements

CSV Data Set Config

CSV file
File name
Comma separated list of variable names
Variables can now be used in requests with ${VariableName}

Requests

${VariableName}

Assertions

^.*${VariableName}.*$

BLAZEMETER

https://www.blazemeter.com/
Functional load testing platform in the cloud
On your local machine, you can scale up to approximately 100 virtual users, but you can go up to 80000 virtual users and 5000 concurrent users with the BlazeMeter pro pricing plan
JMeter script can easily be imported to BlazeMeter and thereby test plans can be executed on a larger scale than locally

Supports Selenium, Taurus, Mocha, Gatling, JUnit, NUnit, Postman and many other types of executors

1. Sign up
2. Create new performance tests
3. Upload JMeter script
4. Set load configuration (Total users / Ramp up time / Duration)
5. Set load distribution and locations
6. Evaluate generated reports with statistics and information on load and response times

BlazeMeter Chrome Plugin
https://chrome.google.com/webstore/detail/blazemeter-the-continuous/mbopgmdnpcbohhpnfglgohlbhfongabi
Can be used to record tests in browser and create test plans

RESOURCES

FUNCTIONAL TESTING
https://www.guru99.com/non-functional-testing.html
https://www.guru99.com/functional-testing-vs-non-functional-testing.html
https://www.guru99.com/what-is-security-testing.html
https://www.guru99.com/reliability-testing.html
https://www.guru99.com/recovery-testing.html
https://www.guru99.com/stability-testing.html
https://www.guru99.com/usability-testing-tutorial.html
https://www.guru99.com/scalability-testing.html
https://www.guru99.com/interoperability-testing.html
https://www.softwaretestinghelp.com/what-is-non-functional-testing/

JMETER
https://jmeter.apache.org/
https://jmeter.apache.org/download_jmeter.cgi
https://jmeter-plugins.org/
http://www.guru99.com/jmeter-tutorials.html
https://www.tutorialspoint.com/jmeter/
https://dzone.com/articles/getting-started-with-jmeter-a-basic-tutorial

BLAZEMETER
https://www.blazemeter.com/