

בבקשה לרשום שם פרטי, שם משפחה, ומספר תעודה זהות בתוך הציגט של הזום.
الرجاء اكتب الاسم الشخصي، اسم العائلة ورقم الهوية، داخل الشات في الزوم.

Напишите пожалуйста ваше имя, фамилию и номер вашего удостоверения личности в чате Зум.

Please type your full name and ID number within the Zoom Chat.

NFT AUTO SW COURSE 2025

Non-Functional Software Testing and Automation with AI

Theory and Practice for Engineers

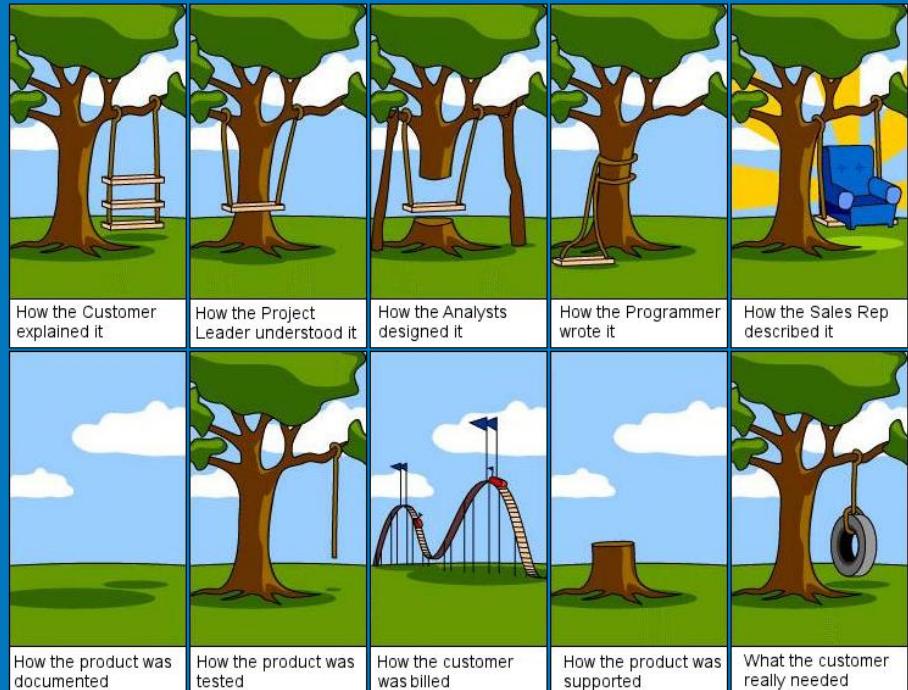
Version 2025.04.06



שלום, עד לסיום חודש יוני, יהיה ניתן לקבוע שעת קבלה בזום איתי בנושא הפרויקט המסייע את הקורס הזה.

NFT Course Units:

1. Introduction to NFT
2. NFT Lab Set-up
3. Test Automation with AI
4. Software Usability
5. Performance and Load Testing - Final Project



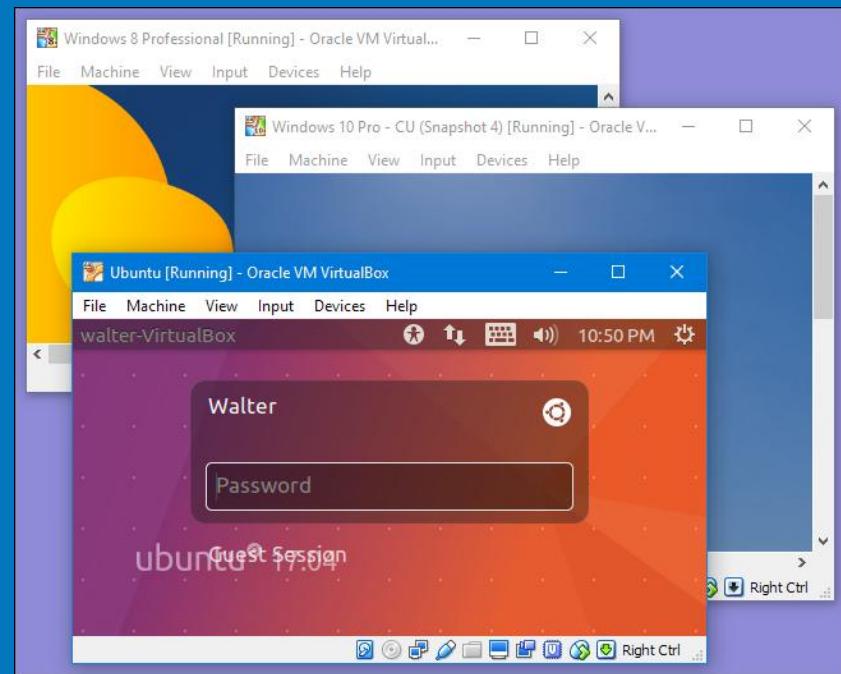
Unit 1: Introduction to NFT

1. Basic Software Testing
2. NFT and Testing Types
3. NFT Software Engineer



Unit 2: NFT Lab

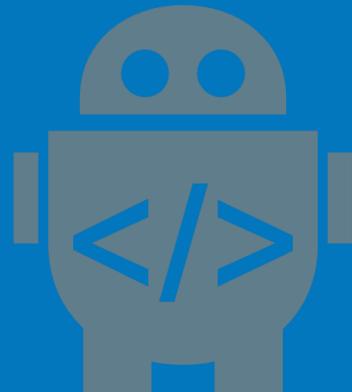
1. Lab Set-up for NFT
2. Virtual Machines for NFT
3. Testing Tools for NFT





Unit 3: Test Automation with AI

1. Introduction
2. Batch files (Windows)
3. Bash scripts (Linux)
4. Postman (Web)



Unit 4: Software Usability

1. Concept
2. Testability



Unit 5: Performance & Load Testing

1. Introduction
2. Black Friday & Cyber Monday
3. Apache JMeter

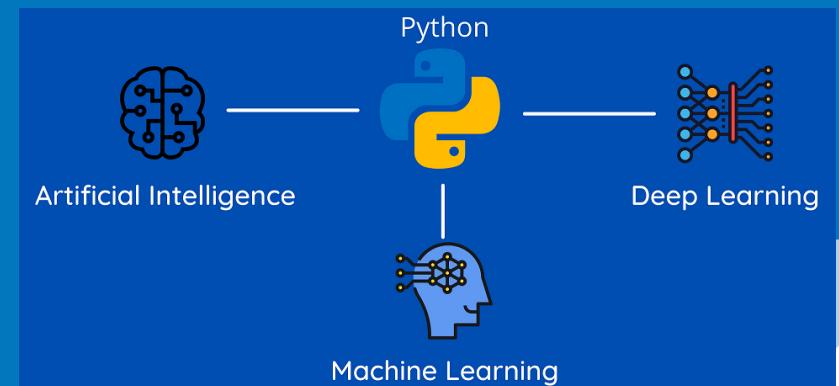


Unit 5: Final Project

Option 1 (DEFAULT): Apache JMeter



Option 2: NFT AI / ML Python



NFT Course Setups:



1. Introduction to NFT - **Setups**: Bugzilla (Web, SQL, Perl), Tor
2. NFT Lab Set-up - **Setups**: Oracle VirtualBox, VMWare Player, Docker, K6
3. Test Automation - **Setups**: Postman, Cursor (AI), VS-Code (AI)
4. Software Usability - **Setups**: N/A
5. Performance and Load Testing - **Setups**: Apache JMeter, Java, Python

WARNING: NO REGISTRY CHANGES ARE REQUIRED !!!

NFT Course Grades:

1. Introduction to NFT - **Labs**: Bugzilla, Tor
2. NFT Lab Set-up - **Labs**: VirtualBox, Regex, K6
3. Test Automation - **Labs**: Batch, Bash, Postman
4. Software Usability - **Labs**: Mobile
5. Performance and Load Testing - **Final Project**: JMeter = 70% (inc. test)



%

Attend. = 10%

5 best = 20%

NFT Course Rules:



- On each assignment or project submission in this course, please add a **CREDITS.TXT** file.
- This file would list the full names and ID numbers of all submitting team members.
- Do not submit the assignment more than once.
- A team leader can be appointed to upload the assignments regularly.
- The ideal size of a team is 3 students.

NFT Course Rules:

- On any assignment or project submission in this course, please do not send any source files of EXE executable of any type.
- All source files must be uploaded to GITHUB and links to the repository must be included in the submission.





Shay Ginsbourg

AI-Powered Software Testing Consultant, Performance & Load Testing Expert, Bio-Med Regulatory Affairs Consultant, & Lecturer at GINSBOURG.CO.IL

Tel Aviv-Yafo, Tel Aviv District, Israel · [Contact info](#)

<https://www.ginsbourg.co.il/>

4,327 followers · 500+ connections



GINSBOURG.CO.IL



Tel Aviv University

from: _____@ac.sce.ac.il



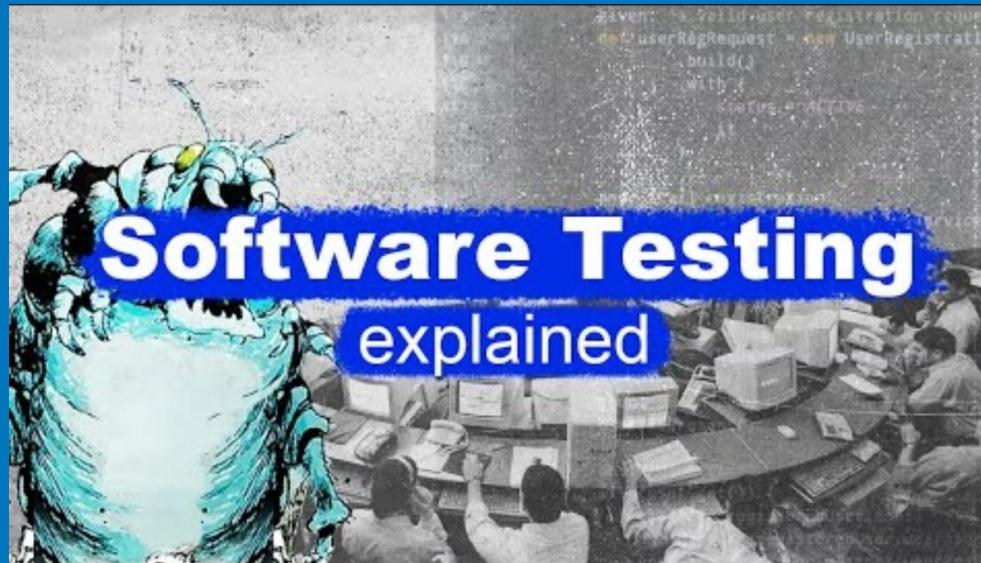
to: shaygi1@ac.sce.ac.il

Unit 1: Introduction to NFT

1. Basic Software Testing
2. NFT and Testing Types
3. NFT Software Engineer

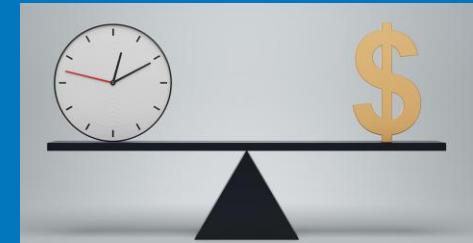


Basic Software Testing (before AI)



<https://youtu.be/oLc9gVM8FBM>

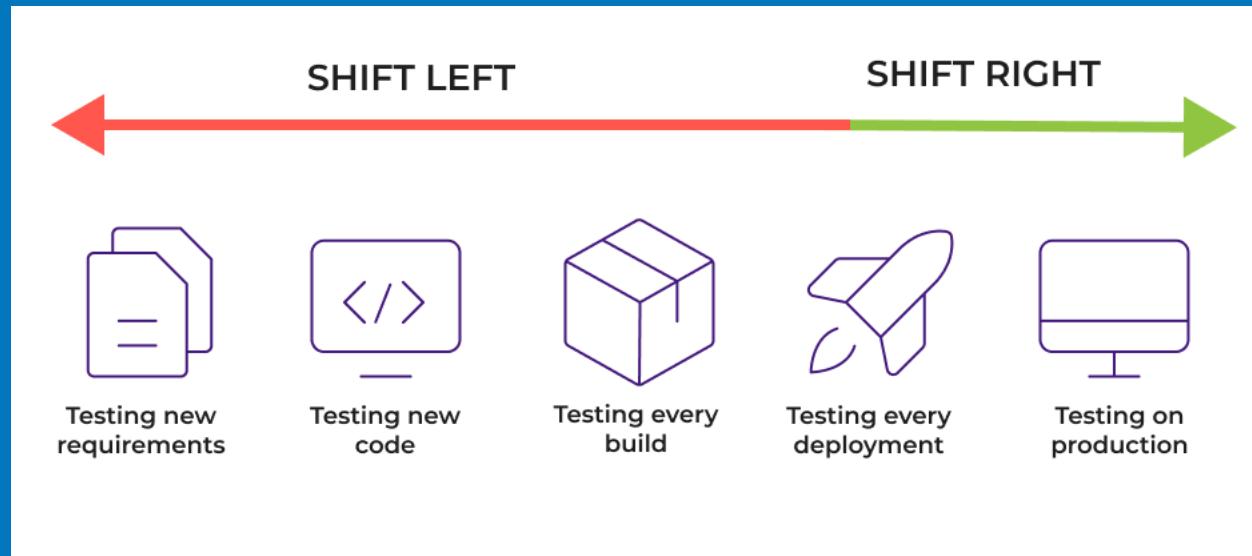
Basic Software Testing



Time is money.

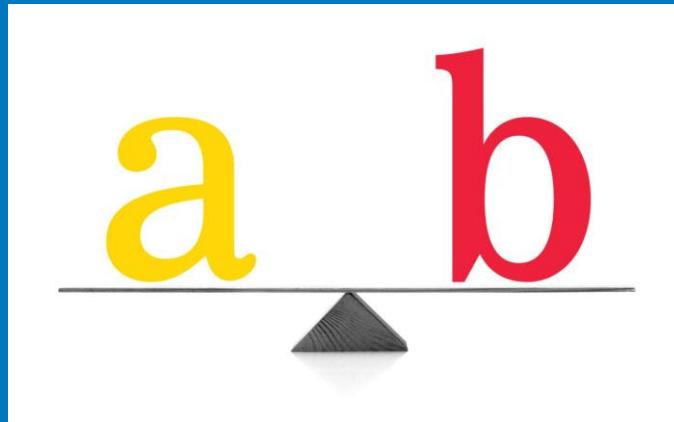
1. Concept
2. R&D
3. Alpha
4. Beta
5. RC
6. GA

Basic Software Testing



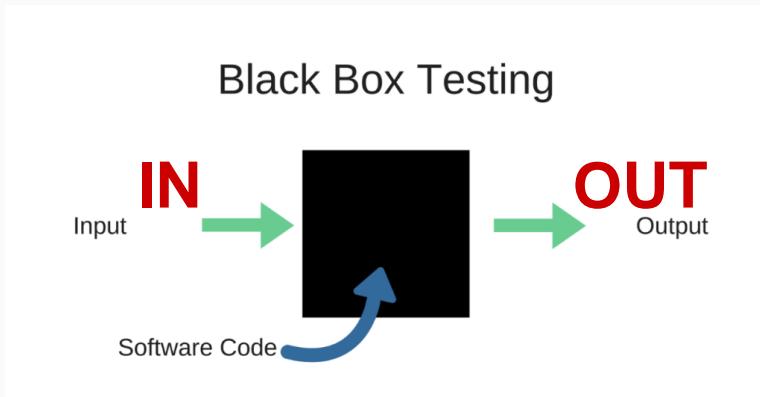
Shift Left is a practice intended to find and prevent defects early in the software development process. The idea is to improve quality by moving tasks to the left as early in the lifecycle as possible.

1.1 Basic Software Testing



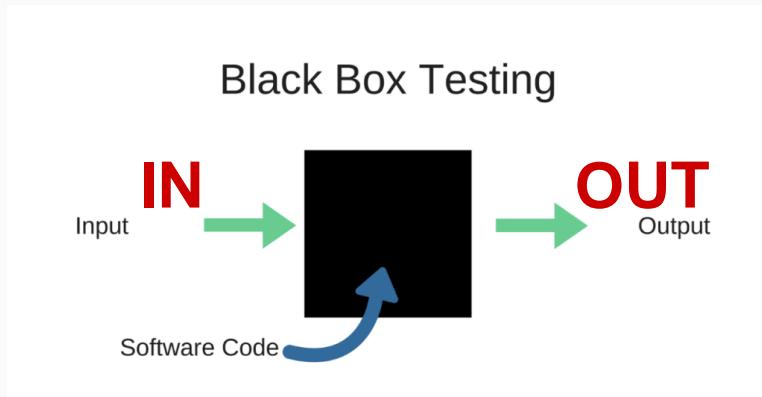
Intuitive A/B testing (sometimes called split testing) is comparing two versions of a web page to see which one performs better.

Black Box Testing



BLACK BOX TESTING, also known as **Behavioral Testing**, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

Black Box Testing

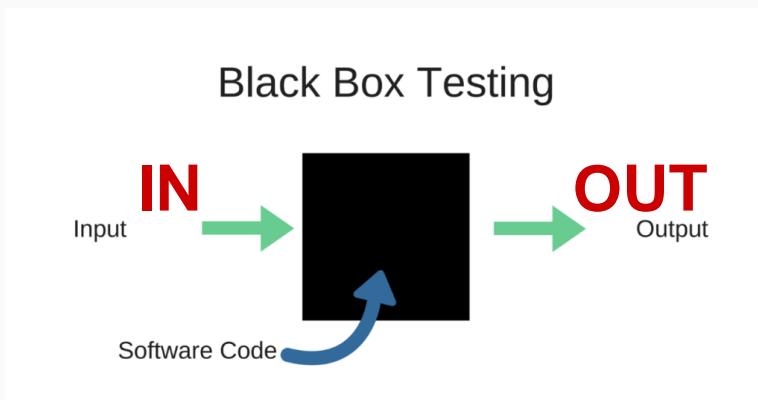


input

1. Normal legal values
2. Abnormal illegal values
3. Error Guessing:
Intuition + Experience
4. Extreme values
5. Equivalent Classes
(borderline values)

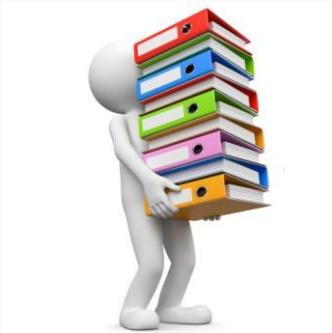
Unit 1: Introduction to NFT

Black Box Testing

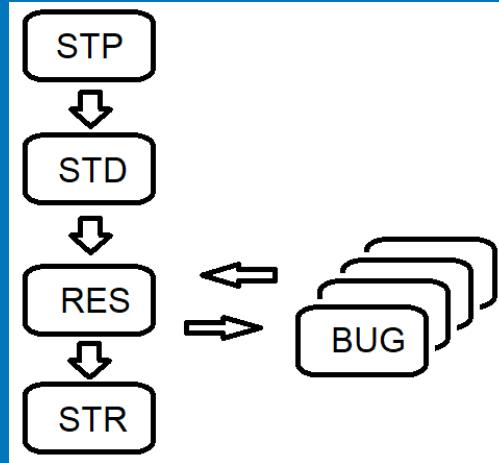


- Easy to try
- Needs only input
- No need to program
- Test conclusion depends too much on Input
- Doesn't check coding efficiency
- Too many combinations

Software QA Documents

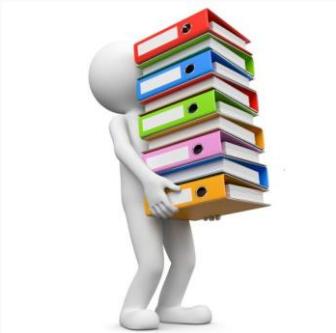


Software Lifecycle Concept



1. Software Test Plan
2. Software Test Description
3. Software Test Bug Reports
4. Software Test Report

Software Test Plan



Software Test Plan (STP) is a management document that describes the scope, approach, resources and schedule of intended test activities. The STP should specify the features to be tested, the testing tasks, who will do each task, and any risks requiring emergency planning. It is a record of the test planning process.

1.1 Basic Software Testing (before AI)

TEST PLAN OUTLINE (IEEE 829 FORMAT)

- 1) Test Plan Identifier
- 2) References
- 3) Introduction
- 4) Test Items
- 5) Software Risk Issues
- 6) Features to be Tested
- 7) Features not to be Tested
- 8) Approach
- 9) Item Pass/Fail Criteria
- 10) Suspension Criteria and Resumption Requirements
- 11) Test Deliverables
- 12) Remaining Test Tasks
- 13) Environmental Needs
- 14) Staffing and Training Needs
- 15) Responsibilities
- 16) Schedule
- 17) Planning Risks and Contingencies
- 18) Approvals
- 19) Glossary

1 TEST PLAN IDENTIFIER

Some type of unique company generated number to identify this test plan, its level and the level of software that it is related to. Preferably the test plan level will be the same as the related software level. The number may also identify whether the test plan is a Master plan, a Level plan, an integration plan or whichever plan level it represents. This is to assist in coordinating software and testware versions within configuration management.



1.1 Basic Software Testing (before AI)

Subject:

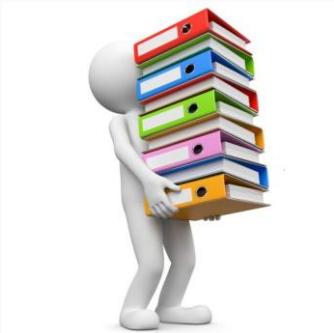
STP (Software Test Plan) for a new web browser
that is better and faster than Chrome and Firefox.

Assignment:

- a. Download a sample software test plan from the Internet.
- b. Use the downloaded plan as a template and create a new test plan to a new unknown web browser.

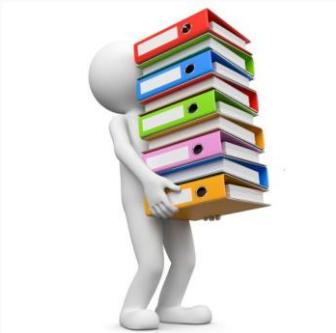


Software Test Description



Software Test Description (STD) is a technical document that describes the test preparations, test cases, test data, and test procedures to be used to perform qualification testing of a software system or subsystem.

Software Test Description



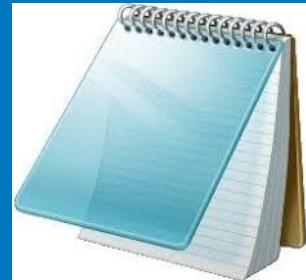
1. Coverage < Requirements
2. Each test focuses on a specific Screen/ Function/ Operation/ Report
3. Black Box rules (negative values, etc.)
4. 5 - 15 test steps per test
5. Independent tests (and data)
6. User point of view

Top Priorities for writing a professional STD:

1. Data Integrity
2. Backwards Compatibility
3. Validity
4. Usability
5. Security



Example STD for Notepad:



1. Data Integrity:

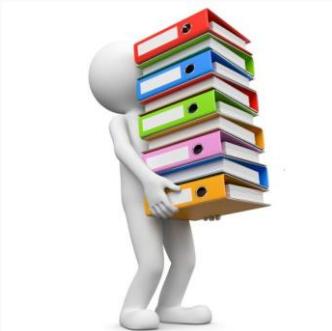
Open Notepad. Create a text document with at least 200 words.

Save as STAM.TXT. Close Notepad. Open again. Check that all the text is exactly the same as was saved.

2. Backwards Compatibility:

Open a old version of Notepad. Create a text document with at least 200 words. Save as STAM.TXT. Close Notepad. Open a new version of Notepad. Check that all the text is exactly the same as was saved.

Software QA Documents



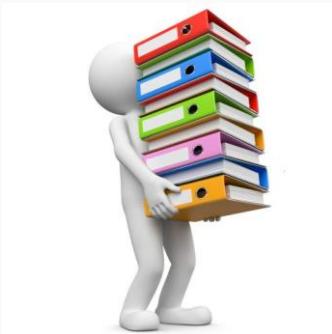
Basic Traceability

Traceability Matrix		
Requirement ID	Test Case ID	Comments
1	1	
2	2 to 5	
3	6	
4	7	
5	8 to 12	
6	13	

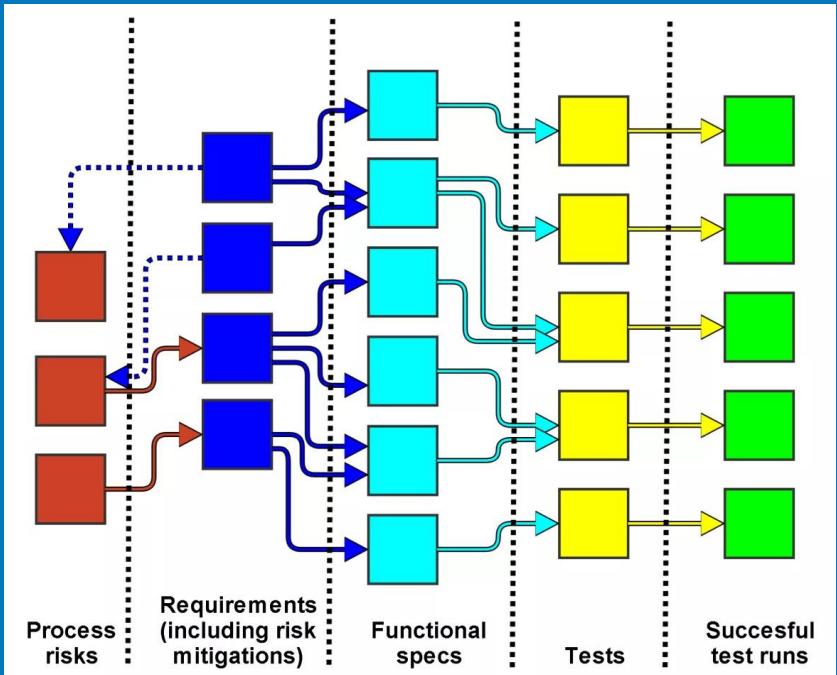


Full Coverage

Software QA Documents



End-to-End Traceability



How to run software tests ?

1. Prepare STD from SRS (Software Requirement Specification) and URS (User Requirement Specification);
2. Run the tests in order;
3. Write the date & time;
4. Write the version number;
5. Write the QA person name.

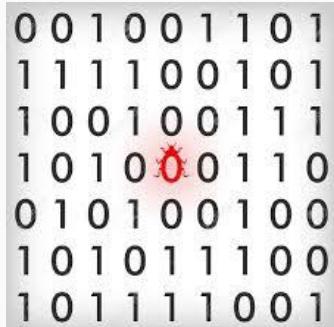


How to run the tests ?

6. Start with Sanity Tests
7. Compare actual to expected results
8. Mark each test step as: Pass /Fail /N.A.
9. If N.A. - then write an explanation
10. If Fail - then open new bugs.



Software Bugs

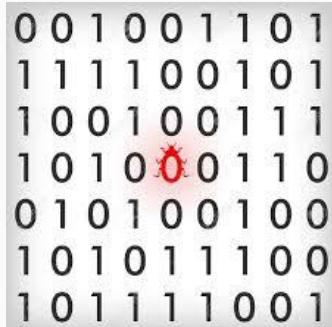


A grid of binary digits (0s and 1s) with a red asterisk highlighting the fourth column of the fourth row.

0	0	1	0	0	1	1	0	1
1	1	1	1	0	0	1	0	1
1	0	0	1	0	0	1	1	1
1	0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0	0
1	0	1	0	1	1	1	0	0
1	0	1	1	1	1	0	0	1

- A **software bug** is an error, flaw, failure or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.

Bug Reporting

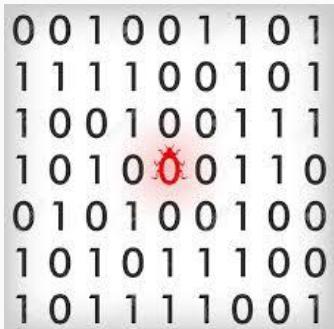


A grid of binary digits (0s and 1s) with a red asterisk (*) highlighting the fourth column from the left, fifth row from the top.

0	0	1	0	0	1	1	0	1
1	1	1	1	0	0	1	0	1
1	0	0	1	0	0	1	1	1
1	0	1	0	0	0	1	1	0
0	1	0	1	0	0	1	0	0
1	0	1	0	1	1	1	0	0
1	0	1	1	1	1	0	0	1

1. Title
2. Reporter
3. Product
4. Version
5. Component
6. Platform
7. Operating system
8. Priority
9. Severity
10. Status
11. Description
12. Attachments

Bug Severity Classification



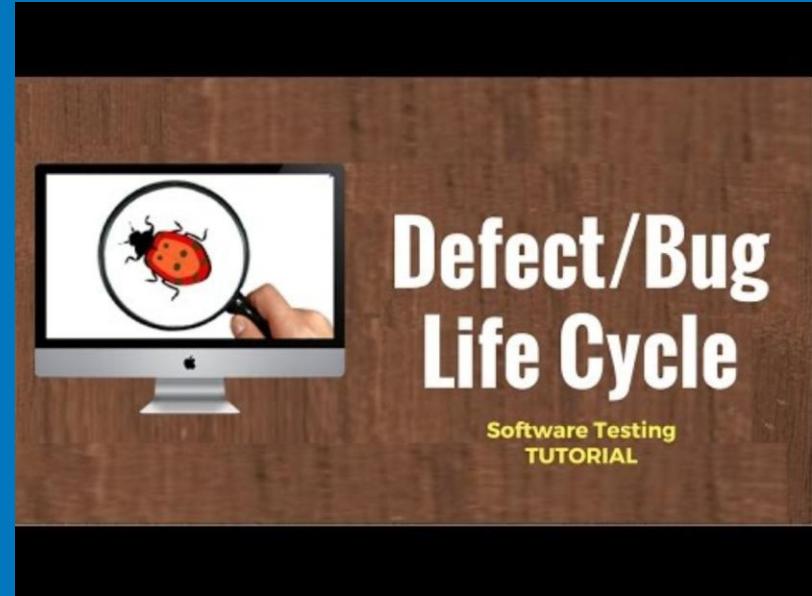
A 7x8 grid of binary digits (0s and 1s). A red asterisk (*) is placed over the 4th digit of the 4th row.

0	0	1	0	0	1	1	0
1	1	1	1	0	0	1	0
1	0	0	1	0	0	1	1
1	0	1	0	0	1	1	0
0	1	0	1	0	0	1	0
1	0	1	0	1	1	1	0
1	0	1	1	1	0	0	1

- **Critical:** The defect affects critical functionality or critical data.
It does not have a workaround.
- **Major:** The defect affects major functionality or major data.
It has a workaround but is not obvious and is difficult.
- **Minor:** The defect affects minor functionality or non-critical data.
It has an easy workaround.
- **Trivial:** The defect does not affect functionality or data. It does not even need a workaround. It does not impact productivity or efficiency.
It is merely an inconvenience.

Bug Life Cycle

```
001001101  
111100101  
100100111  
101000110  
010100100  
101011100  
101111001
```

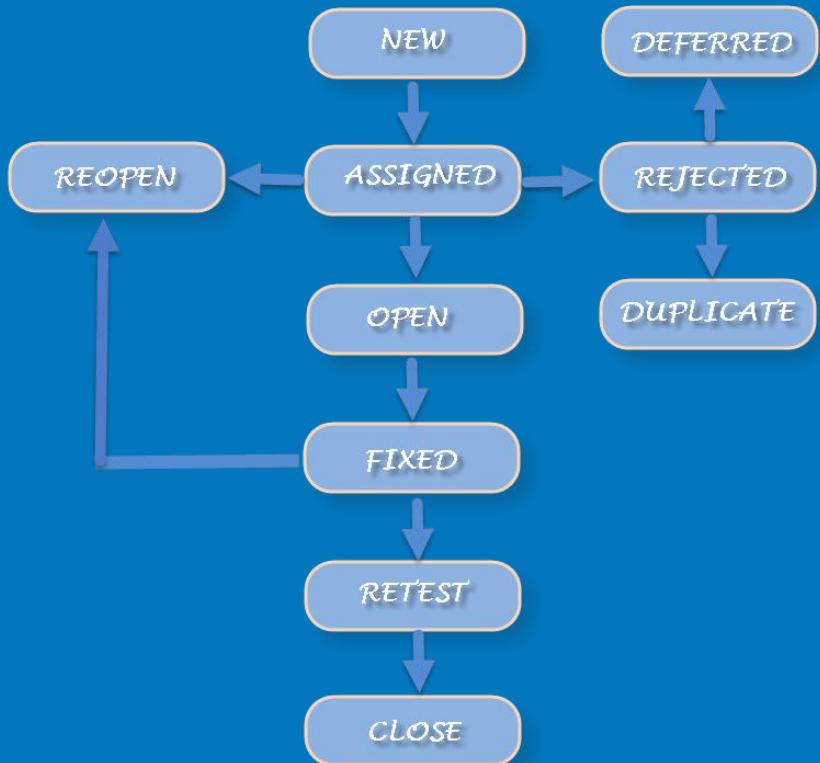


<https://youtu.be/NpDZ2NJmDrE>

Unit 1: Introduction to NFT

Bug Life Cycle

```
001001101  
111100101  
100100111  
101000110  
010100100  
101011100  
101111001
```



How to install and set-up a bug base ?

1. Web Server (Apache, Microsoft IIS)
2. Database Engine (MySQL, PostgreSQL)
3. Perl (ActivePerl, Strawberry Perl)
4. Perl Modules (Email-Sender, Chart)
5. Bugzilla Application



Bugzilla is a web-based general-purpose bugtracker



How to edit a bug in a bug base ?

Screenshot of the Bugzilla bug editing interface:

Bug 39884 - cee27995-8985-4cfe-aa54-a9813cec9e0f (edit)

Status: CONFIRMED (edit)

Alias: None (edit)

Product: MyOwnBadSelf

Component: comp2 (show other bugs)

Version: unspecified

Hardware: All - All

Importance: P2 normal

Target Milestone: ---

Assignee: tara@tequilarista.org (edit) (take)

QA Contact: (edit) (take)

URL:

Whiteboard:

Keywords:

Personal Tags:

Depends on: 38933 (edit)

Blocks:

Show dependency tree / graph

Orig. Est.:	Current Est.:	Hours Worked:	Hours Left:	%Complete:	Gain:	Deadline:
0.0	0.0	0.0 + 0	0.0	0	0.0	<input type="button"/>



Bugzilla is a web-based general-purpose bugtracker



Unit 1: Lab 1

Bugzilla Multi-Layer Architecture:

1. Web Server (Apache, Microsoft IIS)
2. Database Engine (MySQL, PostgreSQL)
3. Perl (ActivePerl, Strawberry Perl)
4. Perl Modules (Email-Sender, Chart)
5. Bugzilla Application



Bugzilla is a web-based general-purpose bugtracker

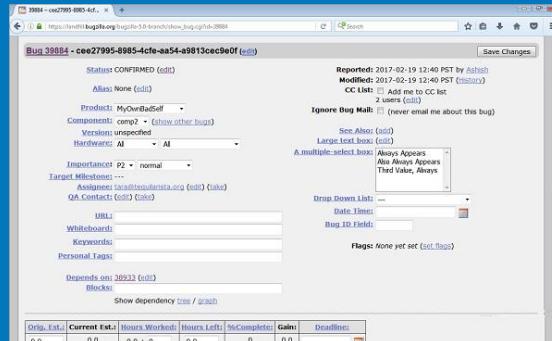


WARNING: NO REGISTRY CHANGES ARE REQUIRED !!!

Unit 1: Lab 1

Tasks:

1. Install all Bugzilla layers,
2. Config and setup Bugzilla,
3. Add users and products,
4. Send email to get points.



Bugzilla is a web-based general-purpose bugtracker



WARNING: NO REGISTRY CHANGES ARE REQUIRED !!!

Unit 1: Lab 1

HELP:

FIND Perl modules live at: <https://metacpan.org>

For each of the modules that you downloaded, complete the following steps:

1. Unpack it into a writable directory.
2. Run the Perl configure command: perl Makefile.pl.
3. Run the make command.
4. Run the make test command. Do not proceed until this command completes successfully.
5. Run the make install command.

CHECK WEB SERVER: <http://localhost>

CHECK BUGZILLA: <http://localhost/bugzilla/>

Installation documentation: <https://bugzilla.readthedocs.io/en/5.2/installing/index.html>



Bugzilla is a web-based general-purpose bugtracker



Alternative solution to BUGZILLA issue with missing PERL modules:



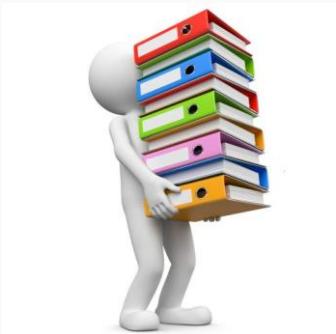
Unit 1: Lab 1

Comment temporary missing PERL libraries in order to launch the application

Bugzilla is a web-based general-purpose bugtracker

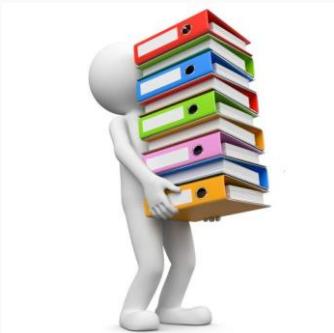


Software Test Report



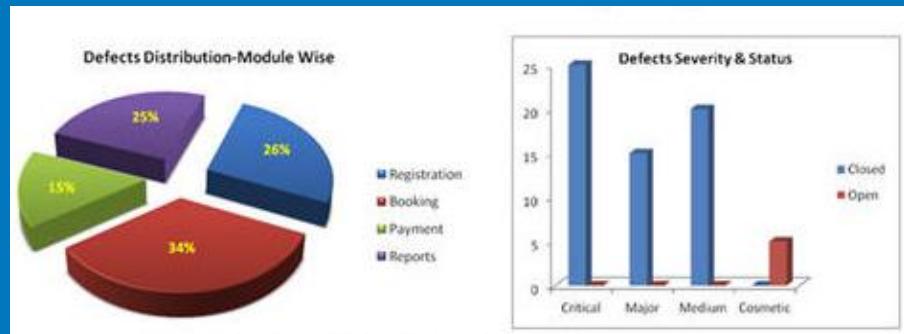
Software Test Report (STR) is a management document that records data obtained from an experiment of evaluation in an organized manner, describes the environmental or operating conditions, and shows the comparison of test results with test objectives.

Software Test Report



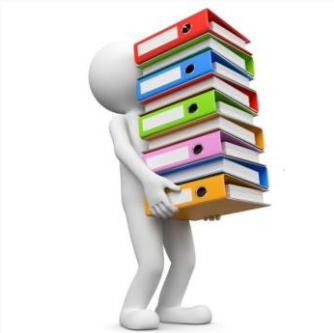
STR = Data + Metadata

Metadata is data about other data



	Registration	Booking	Payment	Reports	Total
Critical	6	7	5	7	25
Major	4	5	2	4	15
Medium	6	8	2	4	20
Cosmetic	1	2	1	1	5
Total-->	17	22	10	16	65

Software Test Report



1. Executive Summary
2. Scope
3. Test environment
4. Software version
5. STD version
6. Full description
7. Data + Metadata
8. Bug highlights

Software Planning and Technical Documentation



<https://youtu.be/2qlcY9LkFik>

NFT and Other Testing Types

1. Sanity Testing
2. Functional Testing
3. Data Integrity Testing
4. Backward Compatibility
5. GUI (Graphical User Interface)
6. Integration Testing
7. Security Testing
8. Disaster Recovery Testing
9. Installation Testing
10. Compatibility Testing
11. Portability Testing
12. Usability Testing
13. Accessibility Testing
14. Regression Testing
15. Load Testing
16. Performance Testing

Testing Types



1. Sanity Testing



Requirement → Testing

Sanity Test example: Smoke Test



Testing Types

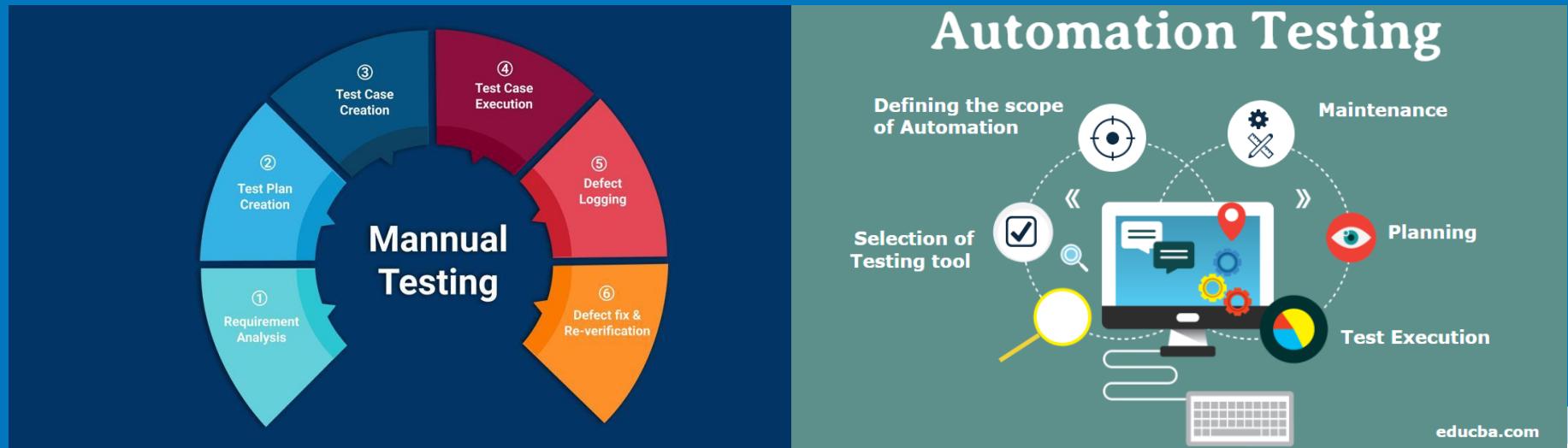


2. Functional Testing



Requirement → Testing

Functional Testing – examples:



Testing Types



3. Data Integrity Testing



Requirement → Testing

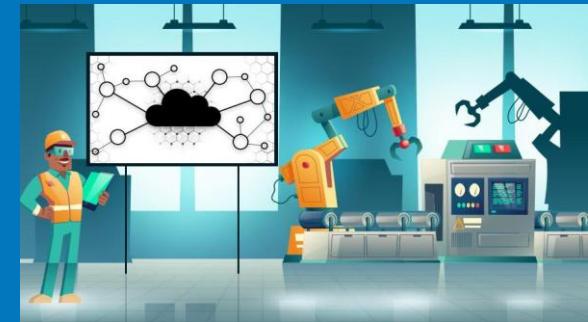
Data Integrity – examples:



Medical Data

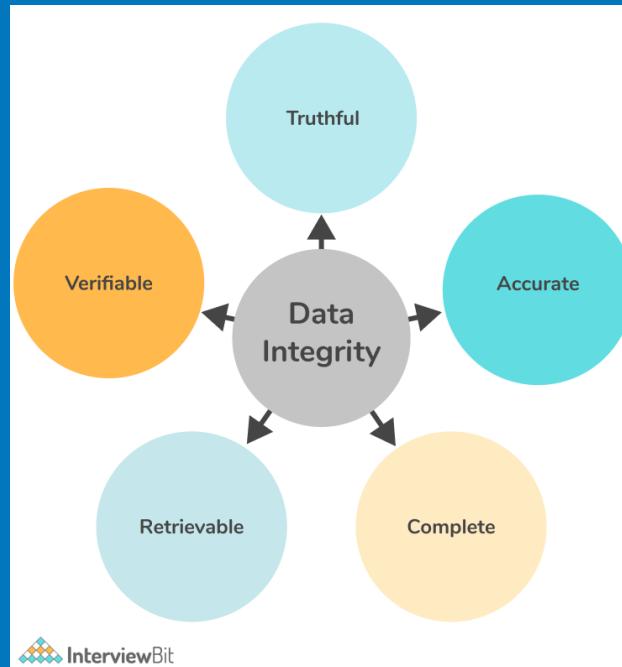


Financial Data



Industrial Data

Data Integrity Testing:



EXAMPLE

Testing Types



4. Backward Compatibility

Disadvantage

No backward compatibility between versions



EXO

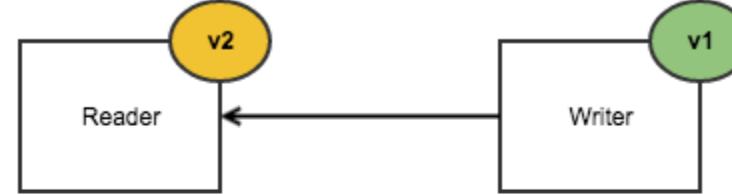
Copyright 2016 InfoPathlet

Requirement → Testing

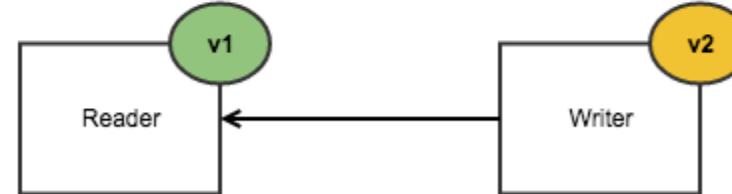
Backward Compatibility Testing:

EXAMPLE

Backwards



Forwards

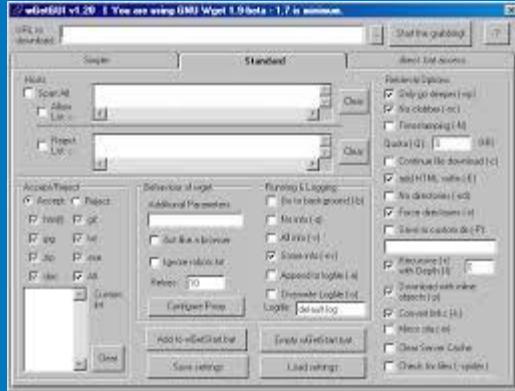


Unit 1: Introduction to NFT

Testing Types

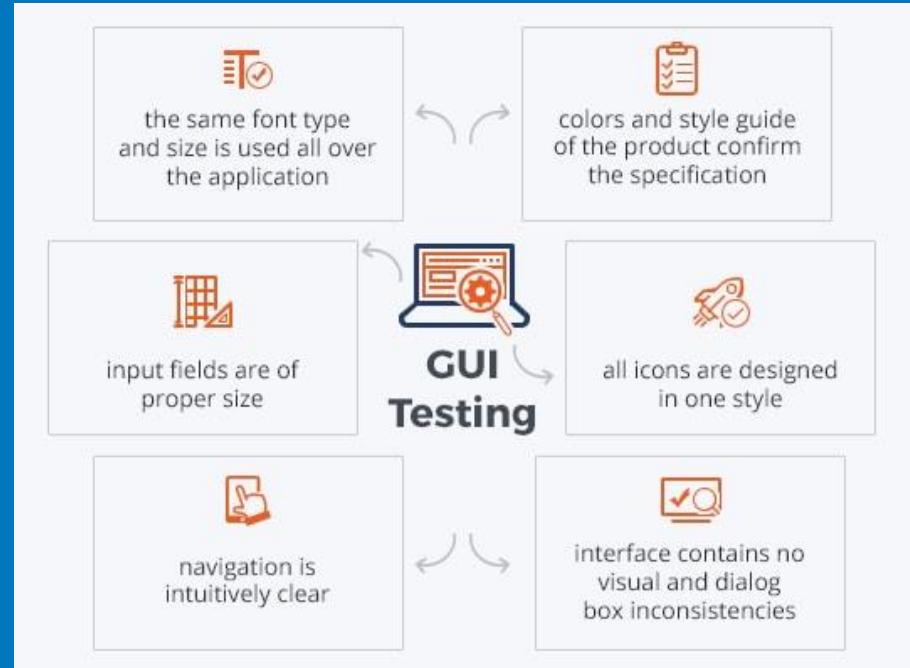
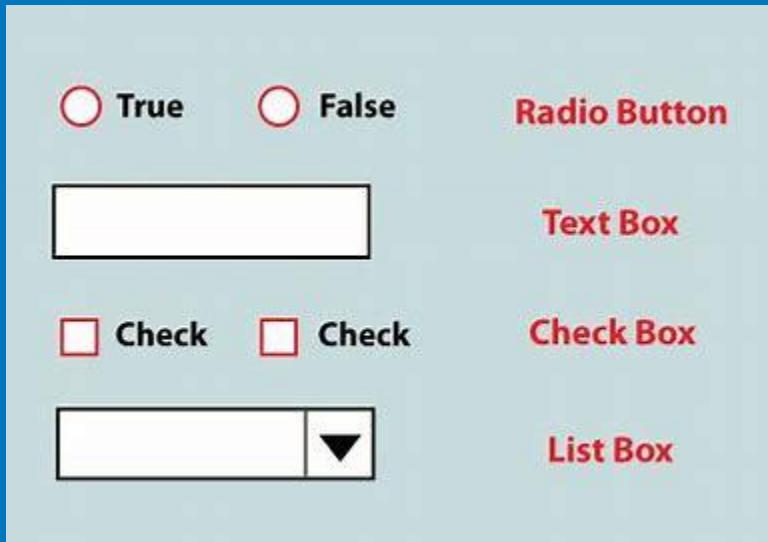


5. GUI (Graphical User Interface)



Requirement → Testing

GUI Testing:

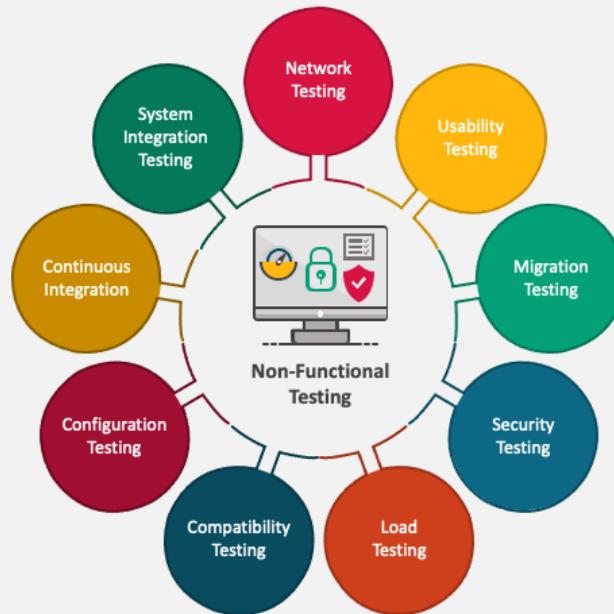


NFT – Start with the requirements:



<https://youtu.be/fc-5HJPBZMQ>

NON-FUNCTIONAL TESTING



Testing Types



6. Integration Testing



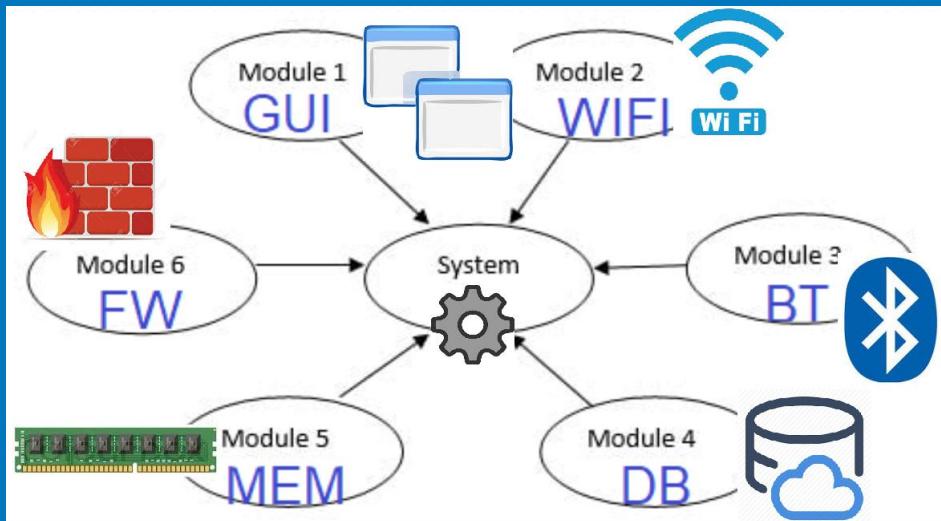
Requirement → Testing

Integration Testing

EXAMPLE



Integration Testing



EXAMPLE

Unit 1: Introduction to NFT

- Security testing — Tests an app's security mechanisms to reveal vulnerabilities.

Testing Types

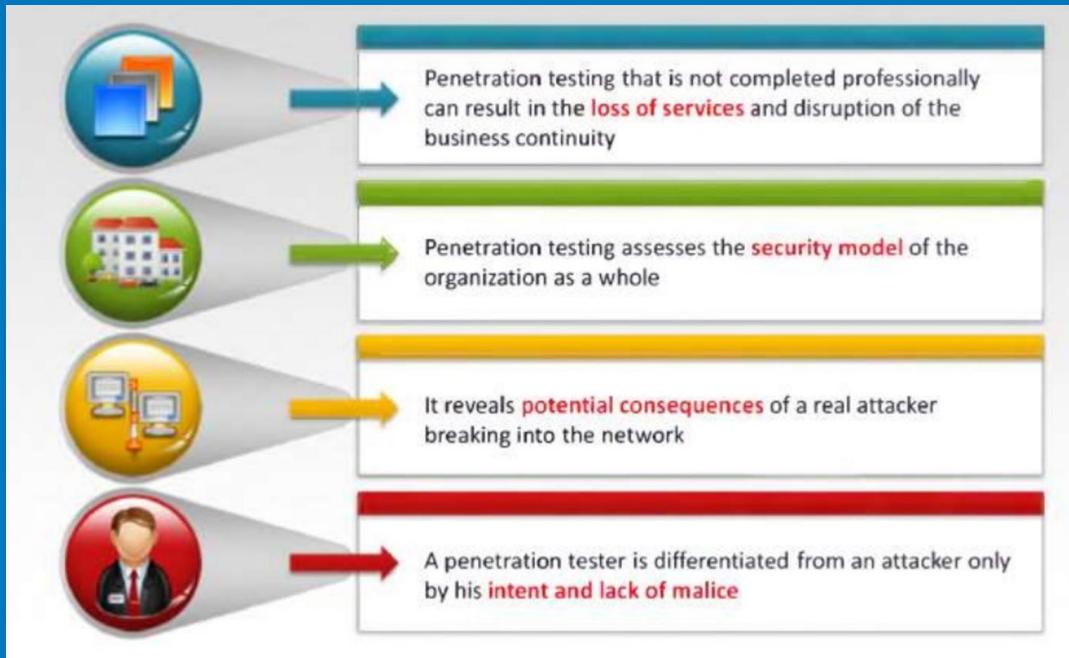


7. Security Testing



Requirement → Testing

Pen Testing



EXAMPLE

Unit 1: Introduction to NFT

- Disaster recovery testing — Tests recovery of business-critical applications in emergency situations.
- Failover testing — Tests an app's backup system in the event of a system failure.

Testing Types

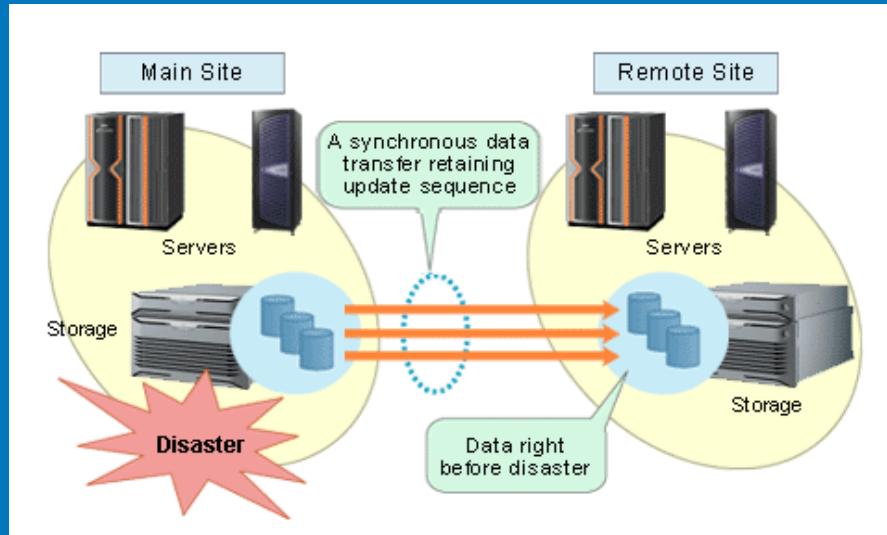


8. Disaster Recovery Testing



Requirement → Testing

Disaster Recovery



EXAMPLE

Minimum
100 miles
(160 kilometers)

ISO 27001/ISO 22301
Disaster Recovery Plan, IT security, and information security

Unit 1: Introduction to NFT

- Installation testing — Tests the initial setup procedure.
- Maintainability testing — Tests the app's ability to update.

Testing Types

- Internationalization testing — Tests if an app can adapt to regional languages and other factors based on location.



9. Installation Testing



Requirement → Testing

Unit 1: Introduction to NFT

- Compatibility testing — Tests the initial setup procedure in various HW/SW environments.
- Compliance testing — Tests whether an app meets specified requirements or regulations.

Testing Types



10. Compatibility Testing



Requirement → Testing

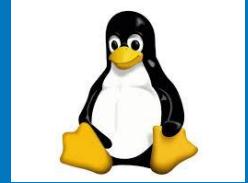
Unit 1: Introduction to NFT

- Portability testing — Tests how an app transfers from one software or operational environment to another.

Testing Types



11. Portability Testing



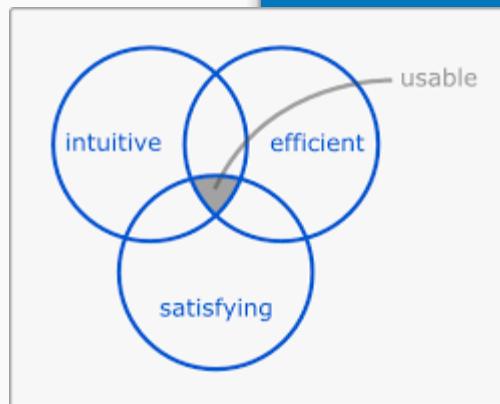
Requirement → Testing

Unit 1: Introduction to NFT

- Usability testing — Tests an app's ease of use.

Usability is the degree to which SW can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use.

Testing Types



12. Usability Testing



Requirement → Testing

Testing Types



12. Usability Testing



Unit 1: Introduction to NFT

- Accessibility testing – Tests how usable the app is to users with special needs, such as vision or hearing impairments.

Testing Types



13. Accessibility Testing

Hear text and controls on the screen
Narrator is a screen reader that reads all the elements on screen, like text and buttons.

Narrator
 On
 Start Narrator automatically
 Off

Voice
Choose a voice

Speed

Magnifier
Use Magnifier to zoom in on parts of your display. Magnifier can run in full screen, in a separate window, or as a lens that follows your mouse pointer around the screen.

Use Magnifier
 On
Press the Windows logo key + Plus (+) to turn on Magnifier.
Press the Windows logo key + Esc to turn off Magnifier.

Make everything on my computer bigger
Change zoom level

On-Screen Keyboard

Esc	~`	! 1 @ 2 # 3 \$ 4 % 5 ^ 6 & 7 * 8 (9) 0 - - + =	Del	Home	PgUp	Nav
Tab	q w e r t y u i o p { [] } \	Del	End	PgDn	Mv Up	
Caps	a s d f g h j k l ; . , Enter		Insert	Pause	Mv Dn	
Shift	z x c v b n m < > ? ^ Shift		PrtScn	ScrLk	Dock	
Fn	Ctrl Alt	Alt Ctrl < > Options	Help	Fade		

Requirement → Testing

Testing Types



Accessibility Testing

vs

Usability Testing



Testing Types



14. Regression Testing



Requirement → Testing

Unit 1: Introduction to NFT

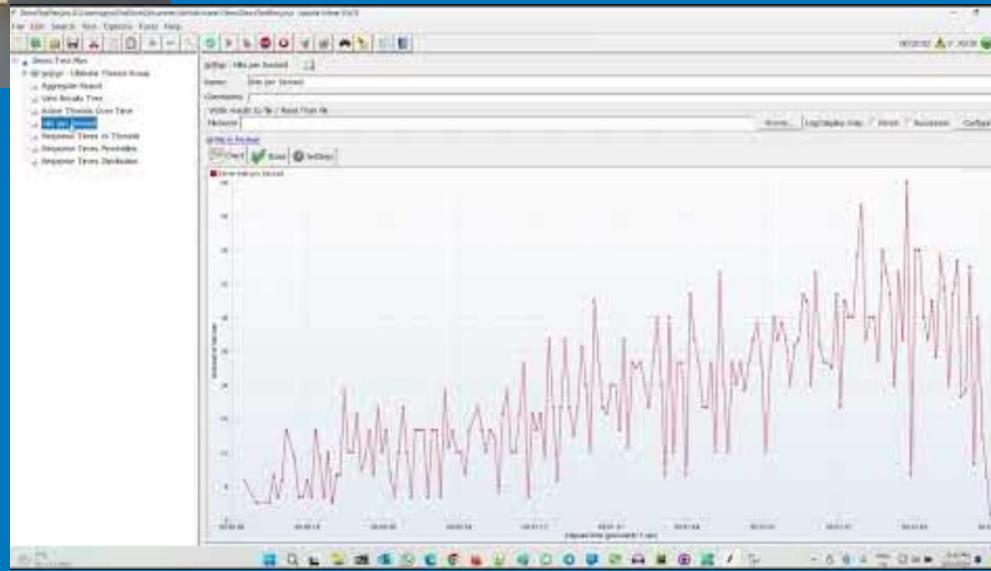
Testing Types



15. Load Testing



Requirement → Testing



EXAMPLE

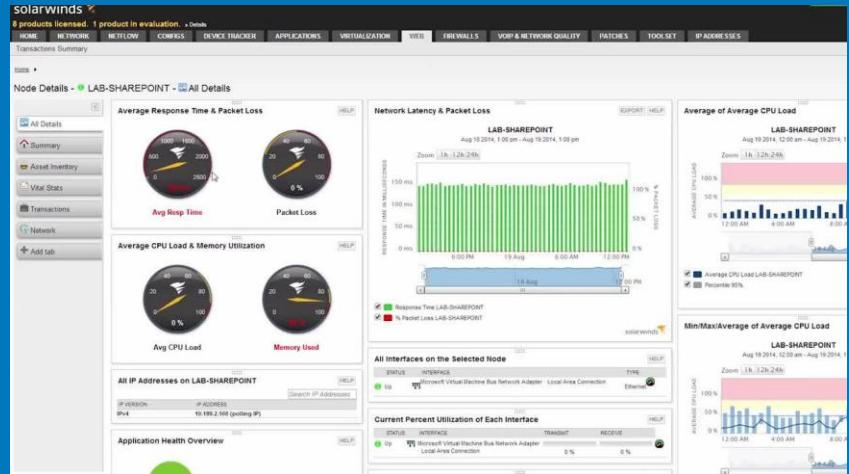
<https://youtu.be/SNZNJj2B2Jw>

Unit 1: Introduction to NFT

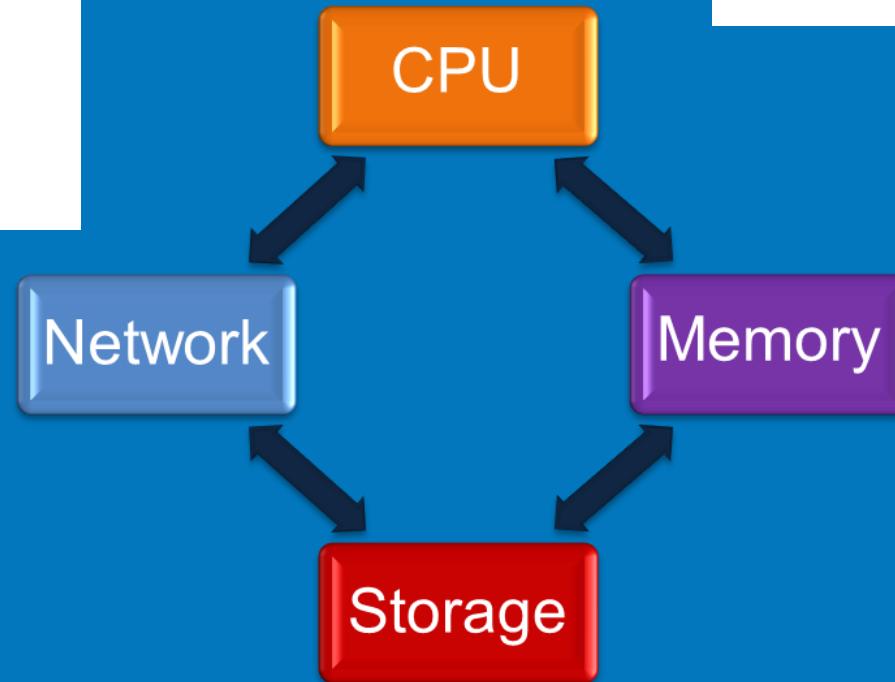
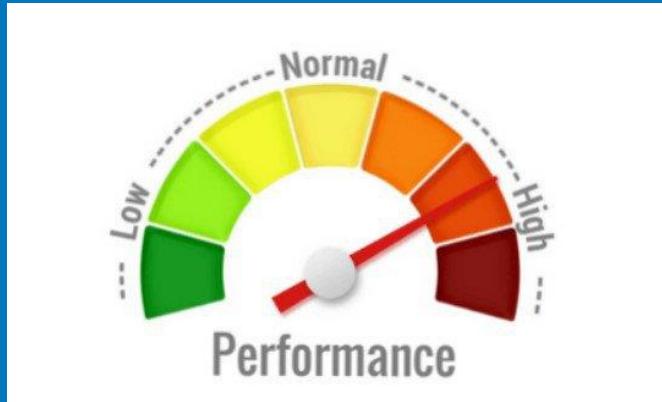
Testing Types



16. Performance Testing



Requirement → Testing



EXAMPLE

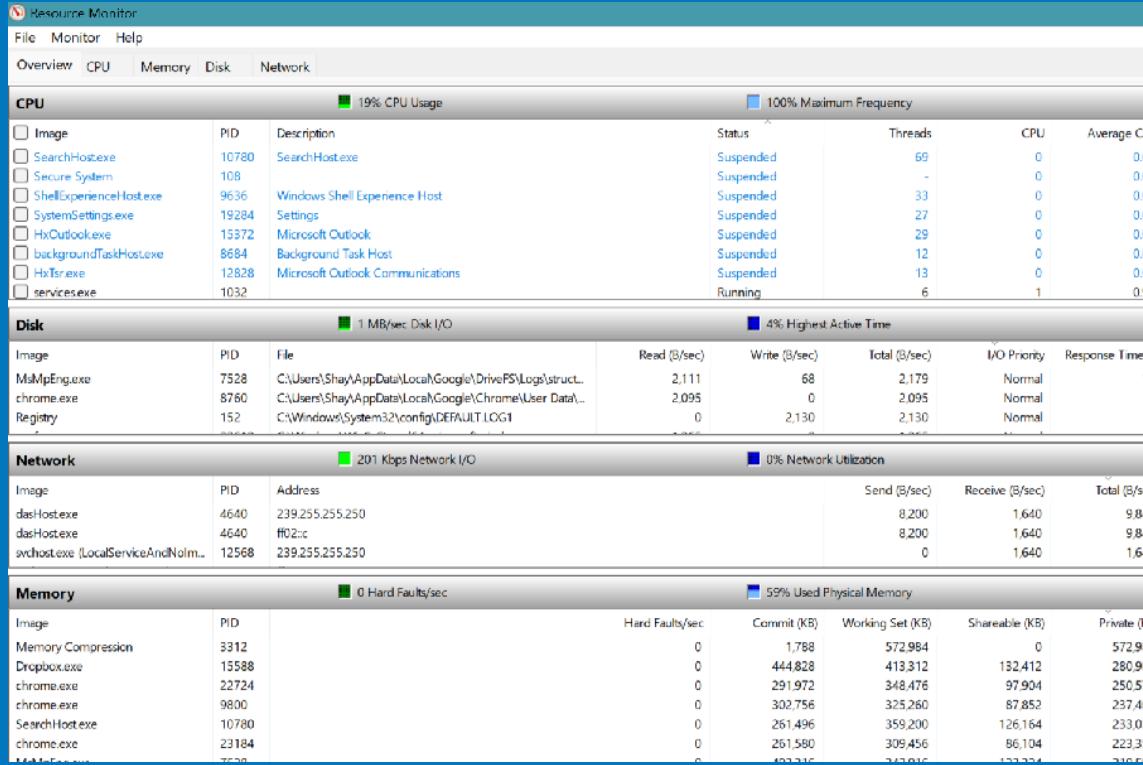
The screenshot shows the Windows Performance Monitor application window. The left sidebar has a tree view under 'Performance' with 'Monitoring Tools' expanded, showing 'Performance Monitor' selected. Below it are 'Data Collector Sets' and 'Reports'. The main pane title is 'Overview of Performance Monitor'. It contains text about using Performance Monitor to view performance data in real time or from a log file, creating Data Collector Sets, and analyzing results. It also describes the new Resource Monitor feature for viewing hardware and system resources. A link to 'Open Resource Monitor' is present. The bottom section is titled 'System Summary' and displays memory and network interface statistics for the computer '\SHAY-ASUS'. The memory summary includes:

	Memory
% Committed Bytes In Use	62.358
Available MBytes	7,024.000
Cache Faults/sec	2,601.416

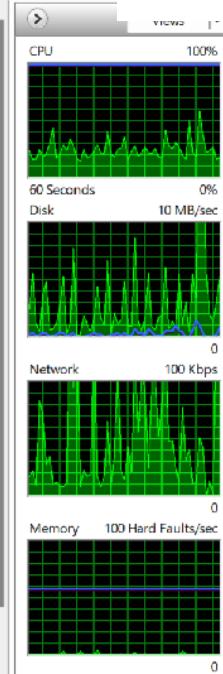
The network interface summary for 'Intel[R] Dual Band Wireless-AC 8265' includes:

	Intel[R] Dual Band Wireless-AC 8265
Bytes Total/sec	8,788.567

A large red stamp with the word 'EXAMPLE' is overlaid across the top right of the main pane.



EXAMPLE



Testing Types



16. Performance Testing

Resilience testing – Tests an app's ability to perform under stressed conditions.

Scalability testing – Tests an app's ability to scale up or down as user requests vary.

Stress testing – Tests an app's stability under heavy loads or extreme conditions.

Load testing – Tests an app's performance under peak conditions.

Endurance testing – Tests an app under a heavy load over an extended period of time.

Performance testing – Tests the speed and responsiveness of an app under various conditions.

Requirement → Testing

Unit 1: Lab 2

According to researchers, only 4% of the internet is visible to the general public. Meaning that the remaining 96% of the internet is made up of “The Deep Web”. Dark Web or Darknet is a subset of the Deep Web where there are sites that sell drugs, hacking software, counterfeit money and more. Accessing hidden marketplace's or darknet websites (with a .onion domain) is done using the TOR network with the TOR browser bundle.
TOR is the most widely used dark web browser.



Unit 1: Lab 2



<https://youtu.be/MUcupYlin68>



Unit 1: Lab 2

Tasks:

1. RESEARCH - Read about TOR and download the software

```
sudo apt update  
sudo apt install torbrowser-launcher
```

2. PLAN - Select 5 NFT types and write test descriptions for TOR
(in a formatted table - see example below)
Focus on testing the browser software - NOT the websites!
3. TEST - Download NFT tools and create tests for TOR



WARNING: NO REGISTRY CHANGES ARE REQUIRED !!!

Unit 1: Lab 2



Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TU01	Check Customer Login with valid Data	Go to site http://demo.guru99.com Enter UserId. Enter Password. Click Submit.	Userid = guru99 Password = pass99	User should Login into an application	As Expected	Pass
TU02	Check Customer Login with invalid Data	Go to site http://demo.guru99.com Enter UserId. Enter Password. Click Submit.	Userid = guru99 Password = glass99	User should not Login into an application	As Expected	Pass

1.3 NFT Software Engineer

1. Social Networks (LinkedIn)
2. Google
3. Tech News
4. Download NFT tools
5. Practice!



1.3 NFT Software Engineer

Subject:

Becoming a pro NFT Software Engineer

Assignment:

- a. Create/Update your LinkedIn profile.
- b. Search Google for NFT software jobs.
- c. Find Tech News websites and register in order to receive weekly email updates.



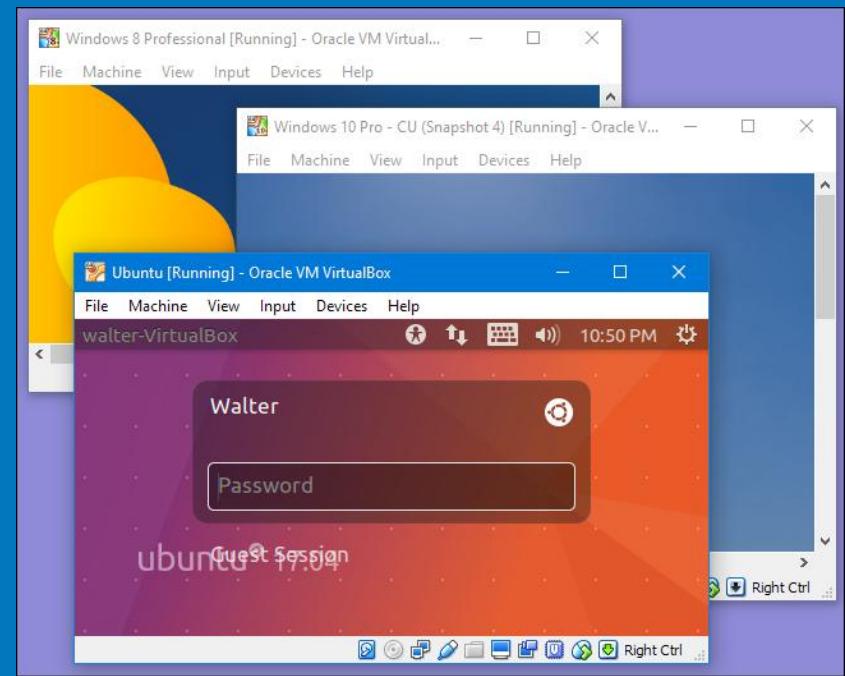
Unit 1: EOF



shaygi1@ac.sce.ac.il

Unit 2: NFT Lab

1. Lab Set-up for NFT
2. Virtual Machines for NFT
3. Testing Tools for NFT



2.1 Lab Set-up for NFT:

If you provide software to clients, you need a plan to test the software for any problems **before deploying** it at the client site.

Ideally, you want the testing lab to simulate the production environment, so **identical equipment** should be purchased for the lab if feasible.

It's also best to **avoid using** the lab testing equipment in the production environment later. Rather, the same equipment can be **used again** in testing sometime in the future.



2.1 Lab Set-up for NFT:

1. Hardware:

Physical / Virtual / Private Cloud / Public Cloud

2. Software:

OS images / Java / Python / Perl

3. Tools:

Test Management Tools - Bugzilla / Filezilla

Test Automation Tools - Selenium / Katalon / Jmeter / Cursor

4. Team:

People skills Vs. Technical skills



2.2 Virtual Machines:

In computing, a virtual machine (VM) is an **emulation** of a computer system.

Virtual machines **provide functionality** of a physical computer.

Their **implementations** may involve specialized hardware, software, or a combination.



2.2 Virtual Machines:

Partial Virtualization:

Wine is a free and open-source **compatibility layer** that aims to allow computer programs (application software and computer games) developed for **Microsoft Windows** to run on **Unix-like operating systems**. Wine provides its own Windows runtime environment which translates Windows system calls into POSIX-compliant system calls. Wine is predominantly written using **black-box testing reverse-engineering**, to avoid copyright issues.



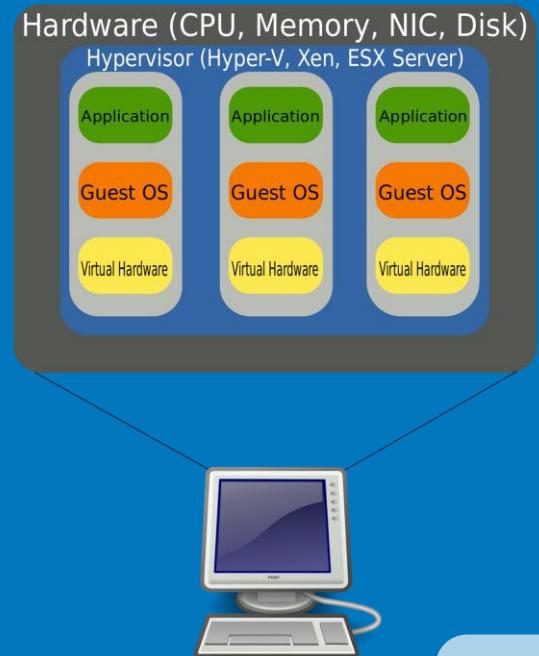
2.2 Virtual Machines:

Full Virtualization:

In full virtualization, the virtual machine simulates enough hardware to allow an unmodified "guest" OS to be run in isolation.

With full virtualization, operating systems and their hosted software are run on top of virtual hardware.

This technology allows multiple guest operating systems to run on a single host OS.



2.2 Virtual Machines:

A **hypervisor** or **Virtual Machine Monitor (VMM)** is computer software, firmware or hardware that creates and runs virtual machines.

A computer on which a hypervisor runs one or more virtual machines is called a **host machine**, and each virtual machine is called a **guest machine**.

Oracle VM VirtualBox is a free and open-source hosted hypervisor for **x86 computers** currently being developed by Oracle Corporation.



2.2 Virtual Machines:

Local Virtual Machines

VMware Workstation Player

Easily run multiple operating systems as virtual machines on your Windows or Linux PC with VMware Workstation Player.

[DOWNLOAD FOR FREE](#)

File	Information
VMware Workstation 17.0.2 Player for Linux 64-bit File size: 498.04 MB File type: bundle Read More	DOWNLOAD NOW
VMware Workstation 17.0.2 Player for Windows 64-bit Operating Systems File size: 577.06 MB File type: exe Read More	DOWNLOAD NOW



VMWARE
WORKSTATION
PLAYER™ 17

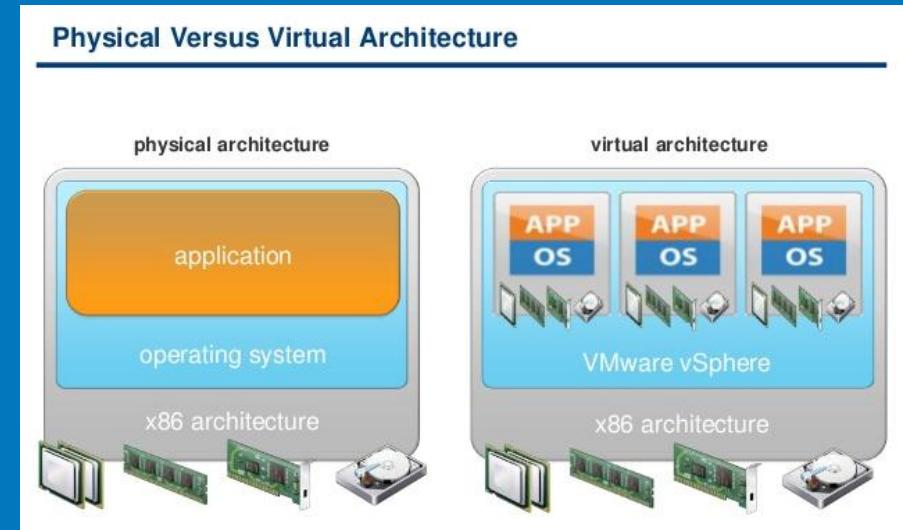
The logo features the VMware logo at the top left, followed by the words "WORKSTATION" and "PLAYER™" in a large, bold, sans-serif font. To the right of "PLAYER™" is a large, stylized number "17" composed of green and teal diagonal bars. The background is white with a large, light blue diagonal shape extending from the bottom left.

100

2.2 Virtual Machines:

The guest can have **access to virtual hardware** that is **less than its host has**: less CPU, less RAM, less DISK SPACE, and less Network bandwidth.

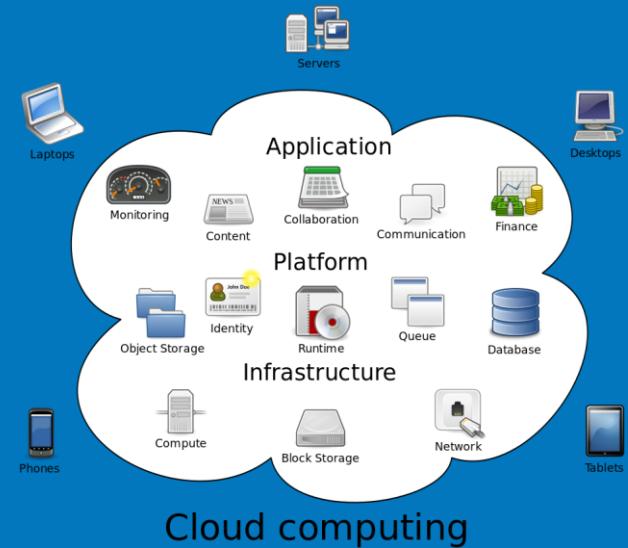
But what is the solution for guests that **need more hardware** than their hosts?



2.2 Virtual Machines:

Cloud Computing makes computer system resources, especially storage and computing power, available on demand without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet.

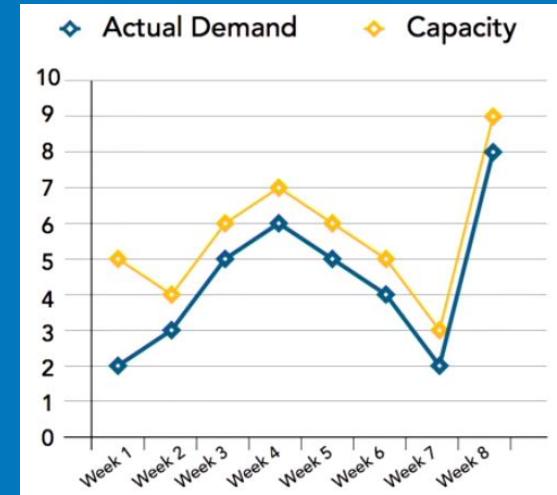
Advantages: Minimize IT costs, get applications up and running faster, improved manageability and less maintenance, and rapidly adjust resources to meet fluctuating and unpredictable demands.



2.2 Virtual Machines:

In cloud computing, **elasticity** is defined as "the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible".

The **dynamic adaptation of capacity**, e.g., by altering the use of computing resources, to meet a **varying workload** is called "elastic computing".



2.2 Virtual Machines:



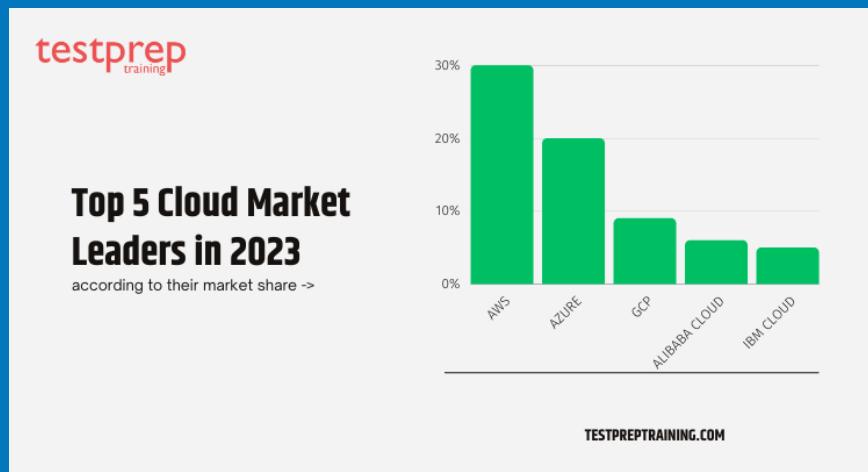
Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios.

Amazon Web Services (AWS) is a secure cloud services platform, offering compute power, database storage, content delivery and more.

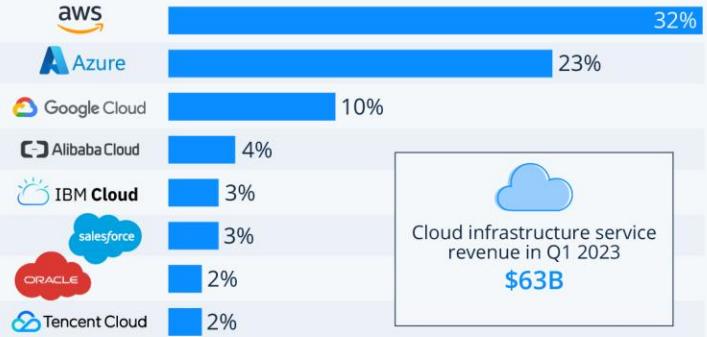


2.2 Virtual Machines:



Big Three Dominate the Global Cloud Market

Worldwide market share of leading cloud infrastructure service providers in Q1 2023*



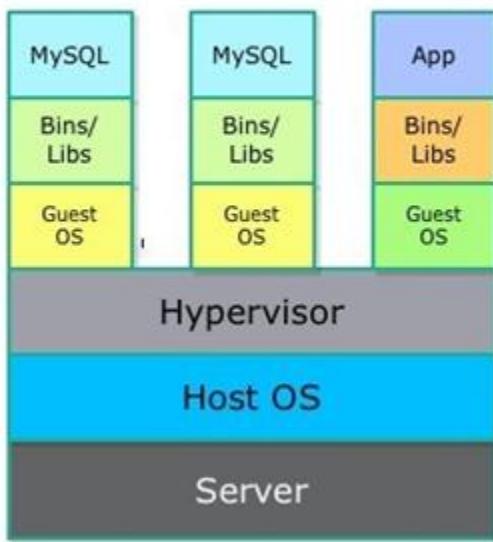
* includes platform as a service (PaaS) and infrastructure as a service (IaaS) as well as hosted private cloud services

Source: Synergy Research Group

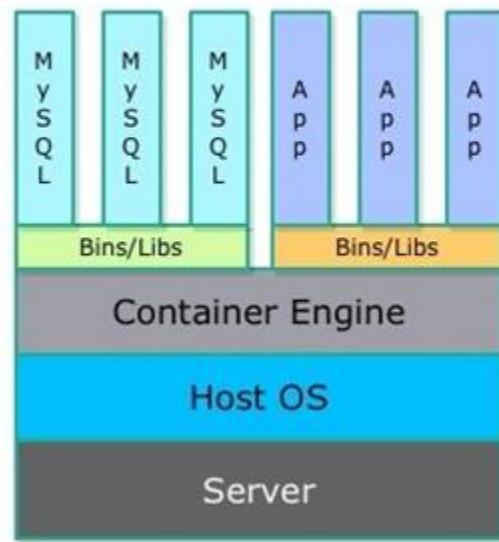


2.2 Virtual Machines:

Virtual Machines



Containers



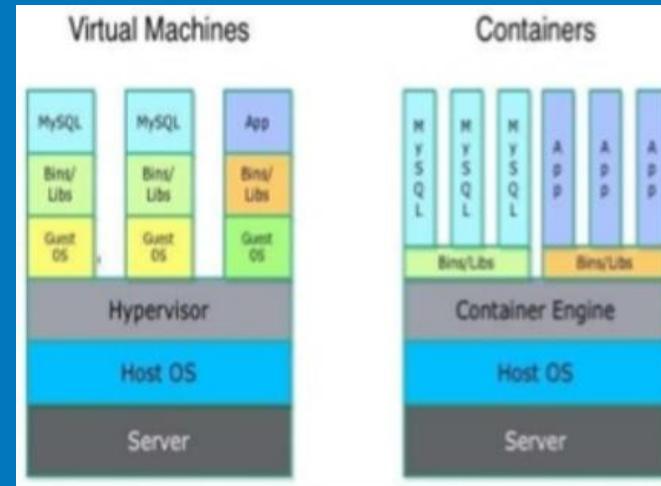
Containers are a common option for deploying and managing software in the cloud. Containers are used to abstract applications from the physical environment in which they are running. A container packages all dependencies related to a software component, and runs them in an isolated environment.

2.2 Virtual Machines:

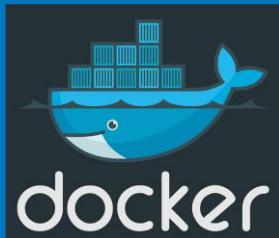
- Unlike a VM, in a container you are not running a complete instance or image of an operating system, with kernels, drivers, and shared libraries.

Instead, an entire stack of containers, whether it be dozens or hundreds or even thousands are able to run on top of a single instance of the host operating system, in a tiny fraction of a footprint of a comparable VM running the same application.

Hypervisor vs Container



2.2 Virtual Machines:



docker create

Create a new container

Usage

```
$ docker create [OPTIONS] IMAGE [COMMAND] [ARG...]
```

Refer to the [options section](#) for an overview of available `OPTIONS` for this command.

Description

The `docker container create` (or shorthand: `docker create`) command creates a new container from the specified image, without starting it.

When creating a container, the docker daemon creates a writeable container layer over the specified image and prepares it for running the specified command. The container ID is then printed to `STDOUT`. This is similar to `docker run -d` except the container is never started. You can then use the `docker container start` (or shorthand: `docker start`) command to start the container at any point.

This is useful when you want to set up a container configuration ahead of time so that it is ready to start when you need it. The initial status of the new container is `created`.

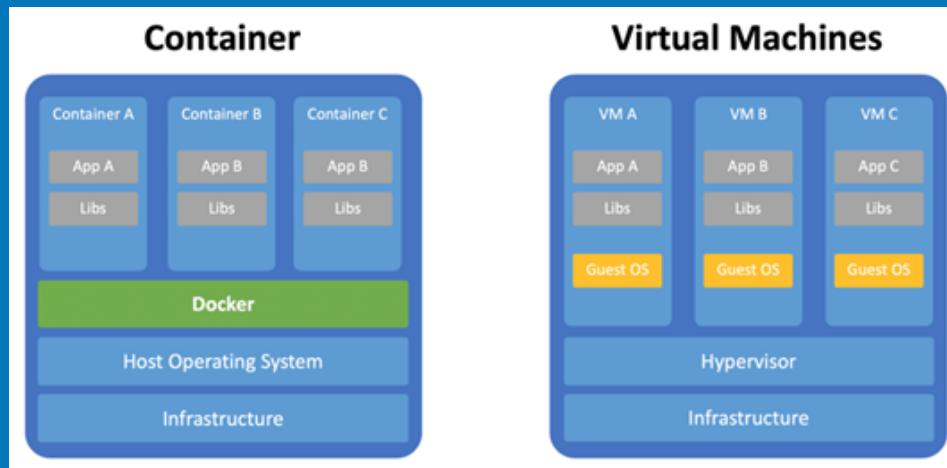
The `docker create` command shares most of its options with the `docker run` command (which performs a `docker create` before starting it). Refer to the `docker run` command section and the [Docker run reference](#) for details on the available flags and options.

← → C docs.docker.com/engine/reference/commandline/create/

 [Search the docs](#) [Home](#) [Guides](#) [Manuals](#) [Reference](#) [FAQ](#) [Samples](#)

[Home](#) / Reference / Command-line reference / Docker CLI (docker) / docker create

2.2 Virtual Machines:



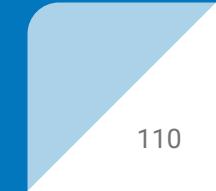
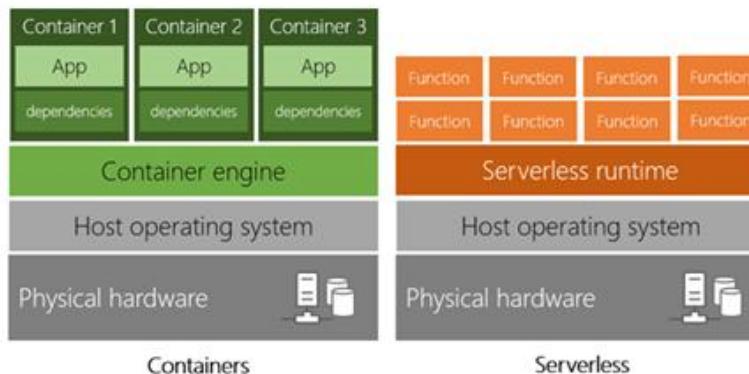
A Docker image is an immutable (unchangeable) file that contains the source code, libraries, dependencies, tools, and other files needed for an application to run.

Due to their read-only quality, these images are sometimes referred to as snapshots. They represent an application and its virtual environment at a specific point in time. This consistency is one of the great features of Docker. It allows developers to test and experiment software in stable, uniform conditions.

A Docker container is a virtualized run-time environment where users can isolate applications from the underlying system. These containers are compact, portable units in which you can start up an application quickly and easily. A Docker container is a virtualized run-time environment where users can isolate applications from the underlying system. These containers are compact, portable units in which you can start up an application quickly and easily.

2.2 Virtual Machines:

Serverless computing is a cloud computing execution model in which the cloud provider allocates machine resources on demand, taking care of the servers on behalf of their customers. Serverless computing does not hold resources in volatile memory; computing is rather done in short bursts with the results persisted to storage. When an app is not in use, there are no computing resources allocated to the app. Pricing is based on the actual amount of resources consumed by an application. It can be a form of utility computing. "Serverless" is a misnomer in the sense that servers are still used by cloud service providers to execute code for developers. However, developers of serverless applications are not concerned with capacity planning, configuration, management, maintenance, fault tolerance, or scaling of containers, VMs, or physical servers.



Unit 2: Lab 3

Oracle Virtualbox Machine, Docker container, and AWS EC2 as a testing lab:

1. Install Oracle Virtualbox (or VMware Player) on Host OS.
2. Create a Virtual Machine with Linux Lite OS (or Ubuntu).
3. Run a sample application on the guest OS.
4. Create a Docker container and run the same sample application on it.
5. Create an account on AWS EC2.
6. Upload the same sample application to a free virtual machine on AWS and run it.



WARNING: NO REGISTRY CHANGES ARE REQUIRED !!!

2.3 Testing Tools:

Test Automation Tools Functions:

1. Script editing/saving
2. Recording capability
3. Functional testing, Regression testing
4. Data integrity
5. Backward compatibility
6. Correlations and Regular expressions
7. Validations
8. Assertions
9. OS Support and Portability
10. Script import/export
11. Parameterization
12. External database access/import
13. Test results saving/export
14. Performance measurements
15. AI/ML libraries



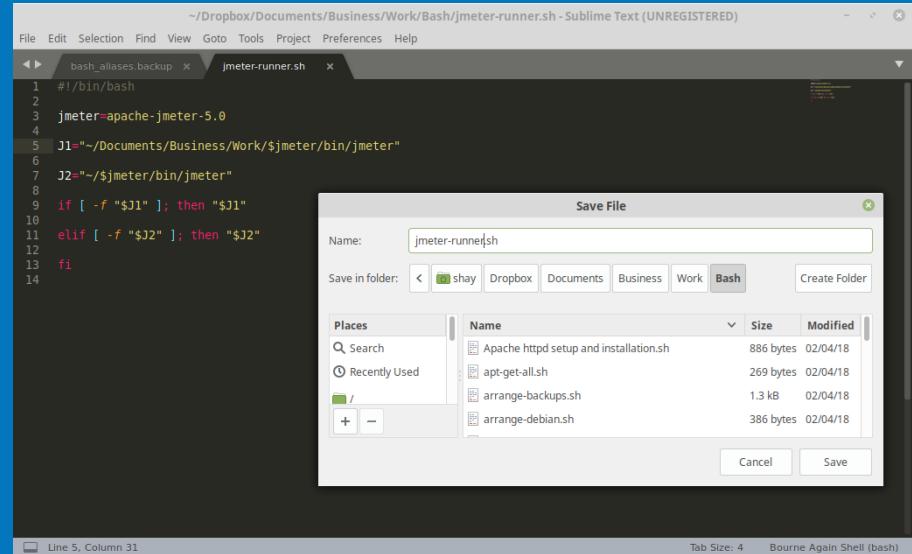
2.3 Testing Tools:

Test Automation Tools Functions:

1. Script editing/saving

Bot generation requires an environment that allows editing and saving the script, which is the “source file” of the runtime bot.

Professional scripting is writing a test using commands and coding knowledge.



The screenshot shows a Sublime Text window with two tabs: 'bash_aliases.backup' and 'jmeter-runner.sh'. The 'jmeter-runner.sh' tab contains the following code:

```
#!/bin/bash
jmeter=apache-jmeter-5.0
J1=~"/Documents/Business/Work/$jmeter/bin/jmeter"
J2=~"/$jmeter/bin/jmeter"
if [ -f "$J1" ]; then "$J1"
elif [ -f "$J2" ]; then "$J2"
fi
```

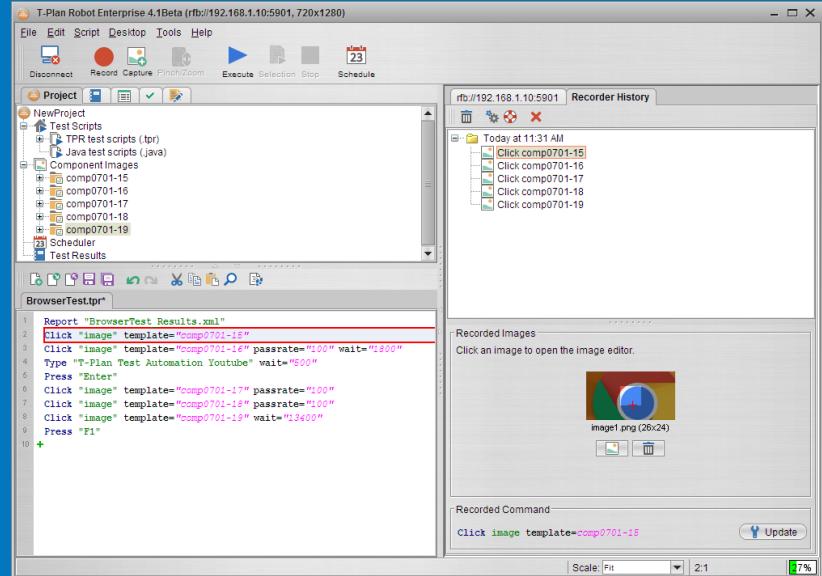
A 'Save File' dialog box is open in the foreground, prompting the user to save the file. The 'Name' field is filled with 'jmeter-runner.sh'. The 'Save in folder' dropdown shows the current path: 'Bash' under 'Dropbox'. The 'Places' sidebar shows recently used files like 'Apache httpd setup and installation.sh', 'apt-get-all.sh', 'arrange-backups.sh', and 'arrange-debian.sh'. The 'Save' button is visible at the bottom right of the dialog.

2.3 Testing Tools:

Test Automation Tools Functions:

2. Recording capability

Record and Replay, otherwise known as **codeless automation**, is a way to run simple tests without programming knowledge. The recorded test may include **session ID**, **time stamps**, and other parameters that are **obsolete** as soon as the recording is completed.



2.3 Testing Tools:

Test Automation Tools Functions:

3. Functional testing, Regression testing

Regression testing is **re-running functional and non-functional tests** to ensure that previously developed and tested software still performs **after being changed**.

Cases that require regression testing include **bug fixes, software enhancements, configuration changes, and even substitution of electronic components**.



2.3 Testing Tools:

Test Automation Tools Functions:

4. Data integrity

Data integrity is the maintenance of, and the assurance of the **accuracy** and **consistency** of, **data over its entire life-cycle**, and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data.



2.3 Testing Tools:

Test Automation Tools Functions:

5. Backward compatibility

Backward compatibility is a property of software systems that allows for interoperability with older legacy systems, or with input designed for such systems.



2.3 Testing Tools:

Test Automation Tools Functions:

/pattern/

6. Correlations and Regular expressions

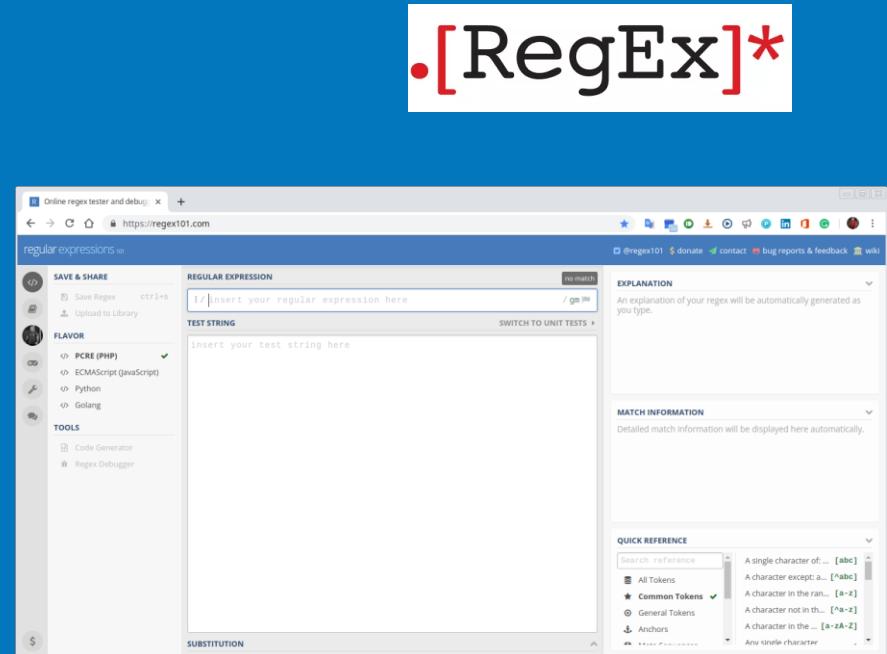
REGEX is a sequence of characters that define a **search pattern**. Usually this pattern is used by string searching algorithms for "**find**" or "**find and replace**" operations on strings, or for input validation. Regular expressions are used in **search engines**, search and replace dialogs of word processors and text editors, and in testing tools.

2.3 Testing Tools:

REGEX for software testing tools:

<https://regex101.com/>

Online regex tester, debugger with highlighting for PHP, PCRE, Python, Golang and JavaScript.



2.3 Testing Tools:

Regex Cheat Sheet

•[RegEx]*

- a Just an 'a' character.
- . Any character except new-line.

- [bgh.] One of the characters listed in the character class b,g,h or . in this case.
- [b-h] The same as [bcdefgh].
- [a-z] Lower case Latin letters.
- [az-] The characters a, z or - (dash).
- [^az] Complementary character class. Anything except a or z.

2.3 Testing Tools:

Regex Cheat Sheet

.[RegEx]*

\w Word characters: [a-zA-Z0-9_]

\d Digits: [0-9]

\s [\f\t\n\r] form-feed, tab, newline, carriage return and SPACE.

\W The complementary of \w: [^\w]

\D [^\d]

\S [^\s]

2.3 Testing Tools:

Regex Cheat Sheet

a?	0-1	'a' characters
a ⁺	1-infinite	'a' characters
a [*]	0-infinite	'a' characters
a{ <i>n,m</i> }	<i>n-m</i>	'a' characters
a{ <i>n,</i> }	<i>n-infinite</i>	'a' characters
a{ <i>n</i> }	<i>n</i>	'a' characters



2.3 Testing Tools:

Regex Cheat Sheet

| Alternation (logical OR)
(...) Grouping and capturing

/i Case insensitive pattern matching

\A Beginning of string
\Z End of string (or before new-line)
\z End of string



2.3 Testing Tools:

/pattern/

REGULAR EXPRESSION

```
// A HREF="(.*?)"
```

TEST STRING

```
<HTML>
<A HREF="tools.html" TARGET=_top>Applications&nbsp;and&nbsp;Languages</A></TD></TR><TR><TD><A
HREF="examples.html" TARGET=_top>Regular&nbsp;Expressions&nbsp;Examples</A></TD></TR><TR><TD><A
HREF="reference.html" TARGET=_top>Regular&nbsp;Expressions&nbsp;Reference</A></TD></TR><TR><TD><A
HREF="refreplace.html" TARGET=_top>Replacement&nbsp;Strings&nbsp;Reference</A></TD></TR><TR><TD><A
HREF="books.html" TARGET=_top>Book&nbsp;Reviews</A></TD></TR><TR><TD><A HREF="print.html"
TARGET=_top>Printable&nbsp;PDF</A></TD></TR><TR><TD><A HREF="about.html"
TARGET=_top>About&nbsp;This&nbsp;Site</A></TD></TR><TR><TD><A HREF="updates.html"
TARGET=_top>RSS&nbsp;Feed&nbsp;&amp;&nbsp;Blog</A></TD></TR></TABLE>
</HTML>
```

8 matches, 324 steps (~1ms) / gm

SWITCH TO UNIT TESTS >

Problem: Found 8 matches instead of specific one

2.3 Testing Tools:



/pattern/

REGULAR EXPRESSION

```
// A HREF=".+?" TARGET=_top>Printable
```

TEST STRING

Fish & Chips

SWITCH TO UNIT TESTS ▾

```
<HTML>
<A HREF="tools.html" TARGET=_top>Applications&nbsp;and&nbsp;Languages</A></TD></TR><TR><TD><A
HREF="examples.html" TARGET=_top>Regular&nbsp;Expressions&nbsp;Examples</A></TD></TR><TR><TD><A
HREF="reference.html" TARGET=_top>Regular&nbsp;Expressions&nbsp;Reference</A></TD></TR><TR><TD><A
HREF="refreplace.html" TARGET=_top>Replacement&nbsp;Strings&nbsp;Reference</A></TD></TR><TR><TD><A
HREF="books.html" TARGET=_top>Book&nbsp;Reviews</A></TD></TR><TR><TD><A HREF="print.html"
TARGET=_top>Printable&nbsp;PDF</A></TD></TR><TR><TD><A HREF="about.html"
TARGET=_top>About&nbsp;This&nbsp;Site</A></TD></TR><TR><TD><A HREF="updates.html"
TARGET=_top>RSS&nbsp;Feed&nbsp;&amp;&nbsp;Blog</A></TD></TR></TABLE>
</HTML>
```

Problem: Found 1 big match instead of specific one

2.3 Testing Tools:



/pattern/

REGULAR EXPRESSION

```
:/ A HREF="([\\w.]+)" TARGET=_top>Printable
```

1 match, 249 steps (~1ms) / gm

TEST STRING

Fish & Chips

SWITCH TO UNIT TESTS ▾

```
<HTML>
<A HREF="tools.html" TARGET=_top>Applications &nbsp;and &nbsp;Languages</A></TD></TR><TD><A
HREF="examples.html" TARGET=_top>Regular &nbsp;Expressions&nbsp;Examples</A></TD></TR><TR><TD><A
HREF="reference.html" TARGET=_top>Regular &nbsp;Expressions&nbsp;Reference</A></TD></TR><TR><TD><A
HREF="refreplace.html" TARGET=_top>Replacement &nbsp;Strings&nbsp;Reference</A></TD></TR><TR><TD><A
HREF="books.html" TARGET=_top>Book &nbsp;Reviews</A></TD></TR><TR><TD><A HREF="print.html"
TARGET=_top>Printable &nbsp;PDF</A></TD></TR><TR><TD><A HREF="about.html"
TARGET=_top>About &nbsp;This &nbsp;Site</A></TD></TR><TR><TD><A HREF="updates.html"
TARGET=_top>RSS &nbsp;Feed&nbsp;&amp;&nbsp;Blog</A></TD></TR></TABLE>
</HTML>
```

No Problem!

2.3 Testing Tools:

•[RegEx]*



https://youtu.be/rhzKDrUiJVk?si=16NM2h9p_A23Dgqu

Unit 2: Lab 4

/pattern/

Create a REGEX for the following patterns:

1. Extract the date from the web page: www.timeanddate.com
2. Extract the IP address from the web page: www.whatismyip.com
3. Extract the main headline from the web page: www.ynet.co.il
4. Extract the download link from: www.bugzilla.org

Unit 2: Lab 4

/pattern/

Create a REGEX for the following patterns:

5. Extract the first 'Tutorial' name from: JMeter.apache.org
6. Extract the names of all NFT course students from: moodle.sce.ac.il
7. Extract 'NFT jobs' from the web page: www.linkedin.com
8. Extract 'software non functional testing' books from: www.amazon.com

2.3 Testing Tools:

Test Automation Tools Functions:

7. **Validations**

Testing tools are required to have features for **validating responses** and ensuring that **accurate and complete data** is provided by the tested software application.

For example, validating HTTP 200 OK success status response code that indicates that the request has succeeded.

Another example, validating that logging into a website was successful and didn't end with HTTP Status-Code 401 (Unauthorized).



2.3 Testing Tools:

Test Automation Tools Functions:

8. Assertions

Testing tools are required to assert or 'break the run' in case that a validation has failed.

For example, do not try to buy a book if login to bookstore has failed.

Another example, do not try to browse a website that sends HTTP status codes that means "service unavailable" and is used by both Apache and Microsoft IIS web servers.



2.3 Testing Tools:

Test Automation Tools Functions:

9. OS Support and Portability

Testing tools are required to support various operating systems, environments, and protocols.

Portability of test scripts is a huge advantage because it allows reusing tests on various environments.



2.3 Testing Tools:

Test Automation Tools Functions:

10. Script import/export

Testing tools are required to import script from other test tools in order to reuse code as much as possible and save development time. Exporting the test script to other formats also increases the total productivity of the testing efforts.



2.3 Testing Tools:

Test Automation Tools Functions:

	A	B	C	D	E	F
1	FROM	TO	AIRLINE	CLASS	PRICE \$	DATE
2	TEL AVIV	ROMA	Turkish	Business	400	23-FEB-2023
3	NEW YORK	LONDON	American	Economy	250	24-FEB-2023
4	BELGRAD	PARIS	Singapore	First	450	25-FEB-2023
5	MALTA	INSTANBUL	Onur	Economy	100	26-FEB-2023

11. Parameterization

Testing tools are required to support parameters. The easiest way to make tests **more realistic and manageable** is to parameterize them, that is, to **replace hard-coded input values** with parameters. This allows reusing the test commands with multiple data sets, share test data and create more **flexible test processes**.

2.3 Testing Tools:

Test Automation Tools Functions:

12. External database access/import

Testing tools are required to connect, access, and/or import information from relevant databases, for example, MS-SQL, PostgreSQL, MySQL, MongoDB, etc.

Testing tools that can't connect to databases should at least be able to import data from spreadsheets or simple text files.



2.3 Testing Tools:

Test Automation Tools Functions:

13. Test results saving/export

Testing tools are required to save their test results in files that can be saved in readable formats and/or exported to readable and editable formats, for example, PDF, HTML etc.



2.3 Testing Tools:

Test Automation Tools Functions:

14. Performance measurements

Performance testing is testing practice executed to determine how a system performs in terms of **responsiveness** and **stability** under a particular workload.

It can also serve to investigate, **measure**, validate or verify other quality attributes of the system, such as **scalability**, **reliability** and **resource usage**.



2.3 Testing Tools:

Test Automation Tools Functions:

14. Performance measurements

Performance testing is testing practice executed to determine how a system performs in terms of **responsiveness** and **stability** under a particular workload.

It can also serve to investigate, **measure**, validate or verify other quality attributes of the system, such as **scalability**, **reliability** and **resource usage**.



2.3 Testing Tools:

Test Automation Tools Functions:

15. AI/ML libraries

Integrating AI/ML into NFT testing can be a **game-changer**. You might want to start by exploring libraries like TensorFlow, PyTorch, or scikit-learn, which are popular for machine learning tasks. They offer tools for data processing, model building, and evaluation that can be applied to test automation and analysis.



Unit 2: Lab 5

VIDEO LESSON 0:

Introduction to performance testing

<https://www.youtube.com/watch?v=Xyq6GItCAvY>



```
import http from 'k6/http';

export default function () {
    http.get('https://test.k6.io');
}
```

WARNING: NO REGISTRY CHANGES ARE REQUIRED !!!

Unit 2: Lab 5

VIDEO LESSON 1:

Frontend vs. Backend performance testing

<https://www.youtube.com/watch?v=xVACRP5qlJI>



```
import http from 'k6/http';
export default function () {
    http.get('https://test.k6.io');
}
```

WARNING: NO REGISTRY CHANGES ARE REQUIRED !!!

Unit 2: Lab 5

VIDEO LESSON 2:

Browser Testing with k6

<https://www.youtube.com/watch?v=N7VJ9X5yAKo>



```
import http from 'k6/http';

export default function () {
    http.get('https://test.k6.io');
}
```

WARNING: NO REGISTRY CHANGES ARE REQUIRED !!!

Unit 2: Lab 5

VIDEO LESSON 3:

Creating k6 scripts with the browser recorder and your favourite IDE

<https://www.youtube.com/watch?v=jEfrw5x2Cbk>



```
import http from 'k6/http';

export default function () {
    http.get('https://test.k6.io');
}
```

WARNING: NO REGISTRY CHANGES ARE REQUIRED !!!

Unit 2: Lab 5



SETUP

Setup

Installation is quite straightforward, although it depends to a degree on the operating system of your machine.

Windows

k6 comes with its own MSI installer for Windows users. Simply download the installer via the link below. Install it normally, and you're good to go.

- [k6 installer](#)

Mac

For Mac users, the easiest method is to run the following command:

```
brew install k6
```

Debian/Ubuntu

Run the following command in your terminal to install it via `apt-get`.

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys C5AD17C747E3415A3642D57D77C6C491D6AC1D69  
echo "deb https://dl.k6.io/deb stable main" | sudo tee /etc/apt/sources.list.d/k6.list  
sudo apt-get update  
sudo apt-get install k6
```

Unit 2: Lab 5



LIFE CYCLE

Test the Life Cycle

In this section, we're going to learn the basic concepts behind k6. Each k6 test consists of four distinct life cycles:

- `init`
- `setup`
- `VU` (virtual user)
- `teardown`

```
1 // 1. initialization code
2
3 export function setup() {
4   // 2. setup code, you can pass data to VU and teardown
5   return {'token': '123456'}
6 }
7
8 export default function (data) {
9   // 3. VU code
10  console.log(data.token);
11 }
12
13 export function teardown(data) {
14   // 4. teardown code
15 }
```

k6-test-life-cycle.js hosted with ❤ by GitHub

[view raw](#)

Unit 2: Lab 5



LIFE CYCLE

k6 requires each test script to contain at least one `default` function, representing the entry point for the `vu` code. `vu` will call the `vu` code over and over again until all conditions are fulfilled.

On the other hand, `init` code will only run once per `vu`. Apart from that, the `setup` and `teardown` cycles will be called only once per test — at the start and end of the whole test life cycle.

You might be wondering about the differences between `init` and `setup`. Unlike `init`, you can actually call all of the k6 APIs, such as HTTP requests, in `setup` and `teardown`. For example, you can make a call in the `setup` code to obtain a token that'll be used inside of the `vu` code.

Unit 2: Lab 5



HTTP REQUESTS

HTTP Requests

HTTP GET request

You can define your first test case by utilizing the built-in `http` module. Create a new JavaScript file in your working directory. I'll just call it `script.js`. Then, add the following code, which represents a simple GET request:

```
import http from 'k6/http';
import { sleep } from 'k6';

export default function () {
    http.get('http://example.com/test');
    sleep(1);
}
```

Each `vu` will execute the code inside the default function sequentially from start to end. Once it has reached the end, it'll loop back, and the process will be repeated all over again.

It's always a good idea to pace out your VUs by adding a `sleep` statement at the end of the default function. This simulates how real users use your system. You can set the value lower to `0.1` for simulating aggressive behaviors. If you intend to simulate users that constantly call your API, simply remove the `sleep` statement.

Unit 2: Lab 5



HTTP REQUESTS

If you don't have an API at the moment and just want to test k6, you can use the following test API provided by k6:

```
http://test.k6.io
```

Run the following code in your terminal:

```
k6 run script.js
```

```
k6 help
```

Unit 2: Lab 5



RESULT OUTPUT

Result output

You should see the following output in your terminal

```
WAKO.io
execution: local
script: script.js
output: - 

scenarios: (100.00%) 1 scenario, 1 max VUs, 10m30s max duration (incl. graceful stop):
  * default: 1 iterations for each of 1 VUs (maxDuration: 10m30s, gracefulStop: 30s)

running (00m01.5s), 0/1 VUs, 1 complete and 0 interrupted iterations
default [=====] 1 VUs 00m01.5s/10m30s 1/1 iters, 1 per VU

  data_received.....: 11 kB 7.2 kB/s
  data_sent.....: 76 B 49 B/s
  http_req_blocked.....: avg=252.89ms min=252.89ms med=252.89ms max=252.89ms p(90)=252.89ms p(95)=252.89ms
  http_req_connecting.....: avg=242.56ms min=242.56ms med=242.56ms max=242.56ms p(90)=242.56ms p(95)=242.56ms
  http_req_duration.....: avg=245.4ms min=245.4ms med=245.4ms max=245.4ms p(90)=245.4ms p(95)=245.4ms
  http_req_receiving.....: avg=997.4μs min=997.4μs med=997.4μs max=997.4μs p(90)=997.4μs p(95)=997.4μs
  http_req_sending.....: avg=943.8μs min=943.8μs med=943.8μs max=943.8μs p(90)=943.8μs p(95)=943.8μs
  http_req_tls_handshaking....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
  http_req_waiting.....: avg=243.46ms min=243.46ms med=243.46ms max=243.46ms p(90)=243.46ms p(95)=243.46ms
  http_reqs.....: 1 0.654912/s
iteration_duration.....: avg=1.49s min=1.49s med=1.49s max=1.49s p(90)=1.49s p(95)=1.49s
iterations.....: 1 0.654912/s
vus.....: 1 min=1 max=1
vus_max.....: 1 min=1 max=1
```

Unit 2: Lab 5



RESULT OUTPUT

By default, k6 will collect the following metrics each time you run a test:

- `vus` — number of active virtual users
- `vus_max` — maximum virtual users allocated for the test
- `iterations` — aggregated number of times the `default` function is called
- `iteration_duration` — the total time it took to execute the `default` function
- `dropped_iterations` — number of `iterations` that couldn't be started
- `data_received` — amount of data received
- `data_sent` — amount of data sent
- `checks` — rate of successful checks (will be discussed later)

In addition, it'll generate the following output as well if you're calling HTTP requests:

- `http_reqs` — total requests generated by k6
- `http_req_blocked` — time spent waiting for a free TCP connection before initiating the request
- `http_req_connecting` — time spent establishing a TCP connection
- `http_req_tls_handshaking` — time spent on TLS handshaking
- `http_req_sending` — time spent on data sending
- `http_req_waiting` — time spent waiting for a response from the remote host
- `http_req_receiving` — time spent on data receiving
- `http_req_duration` — total time for the request. It's calculated based on `http_req_sending + http_req_waiting + http_req_receiving`.

Unit 2: Lab 5



HTTP METHODS

You can specify other arguments as well. Simply run the following to get the full list of the available arguments:

```
k6 help
```

Available HTTP methods

At the time of this writing, k6 comes with the following HTTP methods.

- `batch` — multiple HTTP requests in parallel
- `del` — DELETE request
- `get` — GET request
- `options` — OPTIONS request
- `patch` — PATCH request
- `post` — POST request
- `put` — PUT request
- `request` — issue any type of HTTP request

Unit 2: Lab 5



HTTP POST REQUEST

HTTP POST request

Here's another example of `HTTP POST`, which takes JSON data as its input parameters.

```
1 import http from 'k6/http';
2
3 export default function () {
4     const url = 'http://example.com/login';
5     const payload = JSON.stringify({
6         name: 'wfng',
7         age: '99',
8     });
9
10    const params = {
11        headers: {
12            'Content-Type': 'application/json',
13        },
14    };
15
16    http.post(url, payload, params);
17 }
```

k6-http-post.js hosted with ❤ by GitHub

[view raw](#)

If you are instead testing it on a form-data submission. Simply change the content type to `application/x-www-form-urlencoded`.

Unit 2: Lab 5



OPTIONS

```
import http from 'k6/http';
import { sleep } from 'k6';

export let options = {
  vus: 10,
  duration: '5s',
};

export default function () {
  http.get('http://example.com/test');
  sleep(1);
}
```

Then, you can simply run it normally using the following command:

```
k6 run script.js
```

In fact, you can specify additional arguments when running k6 the command line. For example, the following command will run the test with 10 VUs.

```
k6 run --vus 10 script.js
```

Having said that, you can specify such options in your code as well — providing you with better control over it. Let's use our first example, but this time, we're going to test it with 10 VUs with a duration of five seconds instead.

Unit 2: Lab 5



RAMP UP & DOWN

A good way to test your APIs is to ramp up and down the VU levels. The following code illustrates how you can configure it to run in stages.

```
export let options = {
  stages: [
    { duration: '10s', target: 20 },
    { duration: '1m10s', target: 10 },
    { duration: '10s', target: 0 },
  ],
};
```

- Going from 1 to 20 VUs for the first 10 seconds
- Slowly transitioning to 10 VUs in the next 70 seconds
- Going from 10 to 0 VUs in the last 10 seconds

Unit 2: Lab 5



LOAD TESTING



For load testing, you should ramp up the VU to a good amount and maintain it for a fixed period of time before ramping it down to 0. Have a look at the following example, which uses 100 VUs.

```
export let options = {  
  stages: [  
    { duration: '5m', target: 100 },  
    { duration: '10m', target: 100 },  
    { duration: '5m', target: 0 },  
  ],  
};
```

Can the load test host run your test plan script? Does it have enough CPU? RAM? Network? Can the OS support all the concurrent threads?

Unit 2: Lab 5



STRESS TESTING

On the other hand, stress testing involves the constant ramping up of VUs gradually over a period of time. You can start with 100 VUs and then increment it by 100 VUs each time. Then, you ramp it down as part of the recovery phase.

```
export let options = {
  stages: [
    { duration: '1m', target: 100 },
    { duration: '5m', target: 100 },
    { duration: '1m', target: 200 },
    { duration: '5m', target: 200 },
    { duration: '1m', target: 300 },
    { duration: '5m', target: 300 },
    { duration: '1m', target: 400 },
    { duration: '5m', target: 400 },
    { duration: '5m', target: 0 },
  ],
};
```

Unit 2: Lab 5



SPIKE TESTING

Spike testing aims to overwhelm your system with a sudden surge of a load within a short period of time. You can easily configure it as follows:

```
export let options = {
  stages: [
    { duration: '10s', target: 100 },
    { duration: '2m', target: 100 },
    { duration: '10s', target: 1000 },
    { duration: '2m', target: 1000 },
    { duration: '10s', target: 100 },
    { duration: '2m', target: 100 },
    { duration: '10s', target: 0 },
  ],
};
```

Notice that I've ramped up from 100 to 1,000 VUs within a 10-second time frame. This helps to evaluate the performance of our system when there's a sudden surge of users.

Unit 2: Lab 5



ASSERTIONS: CHECKS

k6 provides a way for you to assert the returned response. It's called `check`. Please note that `check` doesn't halt the execution.

HTTP response

For your information, each HTTP method will return an HTTP response that contains the following:

- `body`
- `headers`
- `status`
- `timings`
- `timings.blocked`
- `timings.connecting`
- `timings.tls_handshaking`
- `timings.sending`
- `timings.waiting`
- `timings.receiving`
- `timings.duration`

Unit 2: Lab 5



ASSERTIONS: CHECKS

Checking if status is 200

The most useful fields are `body` and `status`. You can simply assign the returned HTTP response to a variable and check if its status is `200` as follows:

```
import { check } from 'k6';
import http from 'k6/http';

export default function () {
    let res = http.get('http://example.com/test');
    check(res, {
        'is status 200': (r) => r.status === 200,
    });
}
```

Unit 2: Lab 5



CHECK STATUS 200

Check for HTTP response code

Checks are great for codifying assertions relating to HTTP requests and responses. For example, this snippet makes sure the HTTP response code is a 200:

```
1 import { check } from 'k6';
2 import http from 'k6/http';
3
4 export default function () {
5   const res = http.get('http://test.k6.io/');
6   check(res, {
7     'is status 200': (r) => r.status === 200,
8   });
9 }
```

Unit 2: Lab 5



CHECK RESPONSE TEXT

Check for text in response body

Sometimes, even an HTTP 200 response contains an error message. In these situations, consider adding a check to verify the response body, like this:

```
1 import { check } from 'k6';
2 import http from 'k6/http';
3
4 export default function () {
5   const res = http.get('http://test.k6.io/');
6   check(res, {
7     'verify homepage text': (r) =>
8       r.body.includes('Collection of simple web-pages suitable for load testing'),
9   });
10 }
```

Unit 2: Lab 5



CHECK RESPONSE BODY SIZE

Check for response body size

To verify the size of the response body, you can use a check like this:

```
1 import { check } from 'k6';
2 import http from 'k6/http';
3
4 export default function () {
5   const res = http.get('http://test.k6.io/');
6   check(res, {
7     'body size is 11,105 bytes': (r) => r.body.length == 11105,
8   });
9 }
```

Unit 2: Lab 5



CHECK PASS PERCENTAGE

See percentage of checks that passed

When a script includes checks, the summary report shows how many of the tests' checks passed:

```
$ k6 run script.js
...
✓ is status 200
...
checks.....: 100.00% ✓ 1      X 0
data_received.....: 11 kB    12 kB/s
```

In this example, note that the check "is status 200" succeeded 100% of the times it was called.

Unit 2: Lab 5

Add multiple checks

You can also add multiple checks within a single `check()` statement:



MULTIPLE CHECKS

```
1 import { check } from 'k6';
2 import http from 'k6/http';
3
4 export default function () {
5   const res = http.get('http://test.k6.io/');
6   check(res, {
7     'is status 200': (r) => r.status === 200,
8     'body size is 11,105 bytes': (r) => r.body.length === 11105,
9   });
10 }
```

Unit 2: Lab 5



MULTIPLE CHECKS

When this test executes, the output will look something like this:

```
$ k6 run checks.js

...
✓ is status 200
✓ body size is 11,105 bytes

...
checks.....: 100.00% ✓ 2      X 0
data_received.....: 11 kB    20 kB/s
```

Unit 2: Lab 5



About Failing Checks

When a check fails, the script will continue executing successfully and will not return a 'failed' exit status. If you need the whole test to fail based on the results of a check, you have to [combine checks with thresholds](#). This is particularly useful in specific contexts, such as integrating k6 into your CI pipelines or receiving alerts when scheduling your performance tests.

CHECK FAILS

Unit 2: Lab 5



EXAMPLES

```
// LOAD
export const options = {
  vus: 100,
  duration: '2m',
};
```

```
// SCENARIO
export default function() {

  http.get('https://myapi.com/products/');
  sleep(0.3);

  const data = {username: 'username',
               password: 'password'};
  http.post('https://myapi.com/login/', data);
  sleep(0.3);
}
```

Unit 2: Lab 5



EXAMPLES



```
export const options = {  
  vus: 100,  
  iterations: 200,  
}
```



```
export default function() {  
  http.get('https://myapi.com/products/');  
  sleep(5);  
}
```



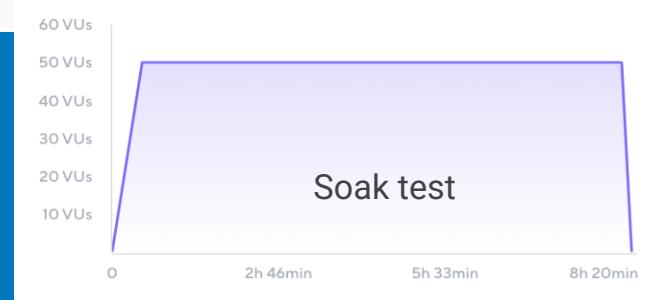
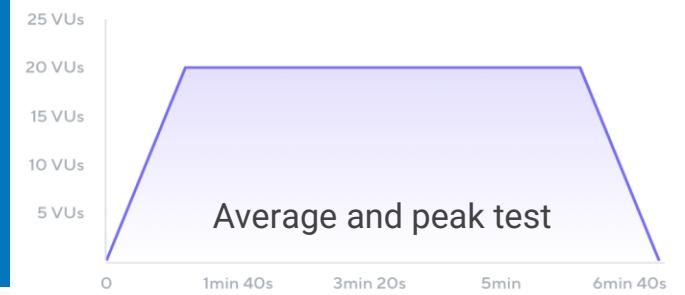
```
export const options = {  
  vus: 100,  
  duration: '30m',  
}
```

Unit 2: Lab 5



LOAD TEST TYPES

```
export const options = {
  stages: [
    { duration: '30s', target: 20 },
    { duration: '1m30s', target: 10 },
    { duration: '20s', target: 0 },
  ],
};
```

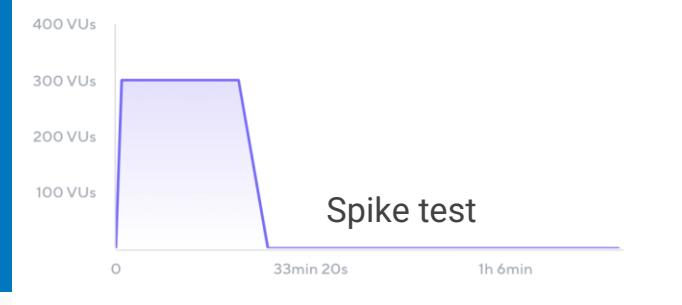


Unit 2: Lab 5



LOAD TEST TYPES

```
export const options = {  
  stages: [  
    { duration: '30s', target: 20 },  
    { duration: '1m30s', target: 10 },  
    { duration: '20s', target: 0 },  
  ],  
};
```



Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Correlation and Dynamic Data

Scripting examples on how to correlate dynamic data in your test script. Correlation is often required when using the Chrome Extension or HAR converter to generate your test script. This is because those tools will capture session IDs, CSRF tokens, VIEWSTATE, wpnonce, and other dynamic values from your specific session. These tokens typically expire very quickly. This is one of the most common things that users will script for when testing user journeys across websites or web apps.

Correlation

In a load testing scenario, correlation means extracting one or more values from the response of one request and then reusing them in subsequent requests. Often, this could be getting a token or some sort of ID necessary to fulfill a sequence of steps in a user journey.

A [recording](#) will capture session data such as CSRF tokens, VIEWSTATES, nonce, etc. This type of data is unlikely to be valid when you run your test, meaning you'll need to handle the extraction of this data from the HTML/form to include it in subsequent requests. This issue is fairly common with any site that has forms and can be handled with a little bit of scripting.

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Extracting values/tokens
from a JSON response

extract-json.js

```
1 import http from 'k6/http';
2 import { check } from 'k6';
3
4 export default function () {
5   // Make a request that returns some JSON data
6   const res = http.get('https://httpbin.test.k6.io/json');
7
8   // Extract data from that JSON data by first parsing it
9   // using a call to "json()" and then accessing properties by
10  // navigating the JSON data as a JS object with dot notation.
11  const slide1 = res.json().slideshow.slides[0];
12  check(slide1, {
13    'slide 1 has correct title': (s) => s.title === 'Wake up to WonderWidgets!',
14    'slide 1 has correct type': (s) => s.type === 'all',
15  });
16
17  // Now we could use the "slide1" variable in subsequent requests...
18 }
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



You can choose from two different approaches when deciding how to handle form submissions. Either you use the higher-level `Response.submitForm([params])` API or you extract necessary hidden fields etc. and build a request yourself and then send it using the appropriate `http.*` family of APIs, like `http.post(url, [body], [params])`.

Extracting values/tokens
from form fields

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Extracting .NET ViewStates, CSRF tokens and other hidden input fields

```
extract-from-hidden.js
1 import http from 'k6/http';
2 import { sleep } from 'k6';
3
4 export default function () {
5   // Request the page containing a form and save the response. This gives you access
6   // to the response object, `res`.
7   const res = http.get('https://test.k6.io/my_messages.php', { responseType: 'text' });
8
9   // Query the HTML for an input field named "redir". We want the value of "redir"
10  const elem = res.html().find('input[name=redir]');
11
12  // Get the value of the attribute "value" and save it to a variable
13  const val = elem.attr('value');
14
15  // Now you can concatenate this extracted value in subsequent requests that require it.
16  // ...
17  // console.log() works when executing k6 scripts locally and is handy for debugging purposes
18  console.log(`The value of the hidden field redir is: ${val}`);
19
20  sleep(1);
21 }
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Sometimes, responses may be neither JSON nor HTML, in which case the above extraction methods would not apply. In these situations, you would likely want to operate directly on the `Response.body` string using a simple function capable of extracting a string at some known location. This is typically achieved by looking for the string "boundaries" immediately before (left) and after (right) the value needing extraction.

The [jslib utils](#) library contains an example of this kind of function, [`findBetween`](#). The function uses the JavaScript built-in [`String.indexOf`](#) and therefore doesn't depend on potentially expensive regular-expression operations.

Generic extraction of
values/tokens

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Extracting a value/token
using `findBetween`

extract-findBetween.js

```
1 import { findBetween } from 'https://jslib.k6.io/k6-utils/1.2.0/index.js';
2 import { check } from 'k6';
3 import http from 'k6/http';
4
5 export default function () {
6   // This request returns XML:
7   const res = http.get('https://httpbin.test.k6.io/xml');
8
9   // Use findBetween to extract the first title encountered:
10  const title = findBetween(res.body, '<title>', '</title>');
11
12  check(title, {
13    'title is correct': (t) => t === 'Wake up to WonderWidgets!',
14  });
15 }
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Correlation

Correlation in k6

Correlation in a load testing scenario refers to retrieving one or more values from one request's response and reusing them in subsequent requests. Obtaining a token or other form of identification is frequently required to complete a series of steps in a user journey.

Why do we use correlation in k6?

When utilizing the Chrome Extension or the HAR converter to produce your test script, correlation is frequently required. This is due to the fact that those tools will record session IDs, CSRF tokens, ViewState, wpnonce, and other dynamic data from your particular session. These tokens usually have a short lifespan. This means you'll have to deal with extracting this data from the HTML/form in order to use it in subsequent requests. This is a fairly common problem with any site that uses forms, and it can be solved with some scripting.

Basically, we have three types of techniques, so let's explore these scripting techniques one by one.

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Correlation

1. Extracting values/tokens from a JSON response

Basically, we use this technique when we have an output response in JSON format, like the below:

```
{"page":2,"per_page":6,"total":12,"total_pages":2,"data":  
  
[{"id":7,"email":"michael.lawson@reqres.in","first_name":"Michael","last_name":  
"Lawson","avatar":"https://reqres.in/img/faces/7-image.jpg"},  
  
 {"id":8,"email":"lindsay.ferguson@reqres.in","first_name":"Lindsay","last_name":  
"Ferguson","avatar":"https://reqres.in/img/faces/8-image.jpg"},  
  
 {"id":9,"email":"tobias.funke@reqres.in","first_name":"Tobias",  
"last_name":"Funke","avatar":"https://reqres.in/img/faces/9-image.jpg"}]
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Correlation

javascript k6 script:-

```
export default function () {  
  
    // Make a request that returns some JSON data  
  
    const res = http.get('https://reqres.in/api/users?page=2');  
  
    // Extract data from that JSON data by first parsing it  
  
    // using a call to "json()" and then accessing properties by  
  
    // navigating the JSON data as a JS object with dot notation.  
  
    const ListUser0 = res.json().data[0];  
  
    check(ListUser0, {  
  
        'first_name in list is correct' : (s) => s.first_name === 'Michael',  
  
        'last_name in list is correct': (s) => s.last_name === 'Lawson',  
  
        'email in list is correct': (s) => s.email === 'michael.lawson@reqres.in'  
    });  
  
    // Now we could use the "ListUser0" variable in subsequent requests...  
}
```



Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Correlation

Related output:-

```
MKG(2).io
execution: local
script: protocol.js
output: - 

scenarios: (100.00%) 1 scenario, 1 max VUs, 10m30s max duration (incl. graceful stop):
  * default: 1 iterations for each of 1 VUs (maxDuration: 10m0s, gracefulStop: 30s)

running (00m00.3s), 0/1 VUs, 1 complete and 0 interrupted iterations
default ✓ [=====] 1 VUs  00m00.3s/10m0s  1/1 iters, 1 per VU

  ✓ first_name in list is correct
  ✓ last_name in list is correct
  ✓ email in list is correct

checks.....: 100.00% ✘ 3      ✘ 6
data_received.....: 4.7 KB 17 KB/s
data_sent.....: 539 B 2.6 kB/s
http_req_blocked.....: avg=200.38ms min=200.38ms med=200.38ms max=200.38ms p(90)=200.38ms p(95)=200.38ms
http_req_connecting.....: avg=62.48ms min=62.48ms med=62.48ms max=62.48ms p(90)=62.48ms p(95)=62.48ms
http_req_duration.....: avg=72.72ms min=72.72ms med=72.72ms max=72.72ms p(90)=72.72ms p(95)=72.72ms
  { expected response:true }.....: avg=72.72ms min=72.72ms med=72.72ms max=72.72ms p(90)=72.72ms p(95)=72.72ms
http_req_failed.....: 0.00% ✘ 0      ✘ 1
http_req_receiving.....: avg=98.86μs min=98.86μs med=98.86μs max=98.86μs p(90)=98.86μs p(95)=98.86μs
http_req_sending.....: avg=4.05ms min=4.05ms med=4.05ms max=4.05ms p(90)=4.05ms p(95)=4.05ms
http_req_tls_handshaking.....: avg=72.83ms min=72.83ms med=72.83ms max=72.83ms p(90)=72.83ms p(95)=72.83ms
http_req_waiting.....: avg=68.56ms min=68.56ms med=68.56ms max=68.56ms p(90)=68.56ms p(95)=68.56ms
http_reqs.....: 1  3.693038/s
iteration duration.....: avg=269.67ms min=269.67ms med=269.67ms max=269.67ms p(90)=269.67ms p(95)=269.67ms
iterations.....: 1  3.693038/s
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Correlations: Extracting Values

2. Extracting values/tokens from form fields

Basically, we use this technique when we have an output response in form fields format like the below:

First name (*)	<input type="text"/>
Last name (*)	<input type="text"/>
Job title	<input type="text"/>
Company (*)	<input type="text"/>
Email (*)	<input type="text"/>
Phone	<input type="text"/>
Message	<input type="text"/>

When we inspect the page or particular element, then we get the below scenario.

```
▼ <input type="text" name="Email" size="55" value maxlength="50"> == $0
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Correlation

Basically, we extract one value in the following fields:

javascript k6 script:-

```
export default function () {  
  
    // Request the page containing a form and save the response. This gives you  
    access  
  
    // to the response object, `res`.  
  
    const res = http.get('https://www.mantissa.co.uk/main/IEForm/contact.php');  
  
    // Query the HTML for an input field named "Firstname", "Email". We want the  
    value of "maxlength" and "size"  
  
    const element1 = res.html().find('input[name=Firstname]');  
  
    const element2 = res.html().find('input[name>Email]');
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Correlation

```
// Get the value of the attribute "value" and save it to a variable.

const val1 = element1.attr('size');

const val2 = element2.attr('maxlength');

// Now you can concatenate this extracted value in subsequent requests that
// require it.

// console.log() works when executing k6 scripts locally and is handy for
// debugging purposes

console.log('The value of field size is: ' + val1);

console.log('The value of name field is:' + val2);

sleep(1);

}
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Correlation

Related output:-

```
INFO[0004] The value of field size is: 30          source=console
INFO[0004] The value of name field is:50          source=console
running (00m05.ls), 0/1 VUs, 1 complete and 0 interrupted iterations
default ✓ [=====] 1 VUs  00m05.ls/10m0s  1/1 iters, 1 per VU
data received.....: 8.0 kB 1.6 kB/s
data sent.....: 651 B 128 B/s
http_req_blocked.....: avg=3.75s min=3.75s med=3.75s max=3.75s p(90)=3.75s p(95)=3.75s
http_req_connecting.....: avg=191.29ms min=191.29ms med=191.29ms max=191.29ms p(90)=191.29ms p(95)=191.29ms
http_req_duration.....: avg=326.36ms min=326.36ms med=326.36ms max=326.36ms p(90)=326.36ms p(95)=326.36ms
  { expected_response:true }.....: avg=326.36ms min=326.36ms med=326.36ms max=326.36ms p(90)=326.36ms p(95)=326.36ms
http_req_failed.....: 0.00% ✓ 0 ✘ x 1
http_req_receiving.....: avg=225.84μs min=225.84μs med=225.84μs max=225.84μs p(90)=225.84μs p(95)=225.84μs
http_req_sending.....: avg=109.28μs min=109.28μs med=109.28μs max=109.28μs p(90)=109.28μs p(95)=109.28μs
http_req_tls_handshaking.....: avg=418.85ms min=418.85ms med=418.85ms max=418.85ms p(90)=418.85ms p(95)=418.85ms
http_req_waiting.....: avg=326.02ms min=326.02ms med=326.02ms max=326.02ms p(90)=326.02ms p(95)=326.02ms
http_reqs.....: 1  0 196849/x
iteration_duration.....: avg=5.07s min=5.07s med=5.07s max=5.07s p(90)=5.07s p(95)=5.07s
iterations.....: 1  0 196849/x
vus.....: 1  min=1  max=1
vus_max.....: 1  min=1  max=1
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



Generic extraction of values/tokens

We use this technique when responses may be neither JSON nor HTML at times, in which case the above extraction methods would not be applicable. In these cases, you should probably work directly on the Response. we can handle this type of situation by using **FindBetween** inbuilt function.

The jslib utils library contains an example of this kind of function, called **findBetween**. The function makes use of JavaScript's built-in String.indexOf and therefore does not depend on the use of potentially expensive regular expression operations.

Let's take the XML response example for a better understanding

Correlation

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



EXPECTED RESULTS

```
<travelers>

<Travelerinformation>

<id>1403</id>

<name>Mayank Narayana</name>

<email>mayank-leo@hotmail.com</email>

<addreses>USA</addreses>

<createdat>2020-12-16T09:57:10.9389781</createdat>

</Travelerinformation>

</travelers>
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



EXPECTED RESULTS

javascript k6 script:-

```
import { findBetween } from 'https://jslib.k6.io/k6-utils/1.2.0/index.js';

export default function () {

    // This request returns XML:

    const res = http.get('http://restapi.adequateshop.com/api/Traveler?
page=6');

    // Use findBetween to extract the first title encountered:

    const name = findBetween(res.body, '<name>','</name>');

    const email= findBetween(res.body, '<email>','</email>');
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5



EXPECTED RESULTS

```
check(name, {  
  
    'name is correct': (ast) => ast === 'Mayank Narayana',  
  
});  
check(email, {  
  
    'email is correct': (ast) => ast === 'mayank-leo@hotmail.com',  
  
});  
console.log('The value of name field is:' + name);  
  
console.log('The value of name field is:' + email);  
}
```

Correlation is retrieving a value from a prior HTTP response and using it in a further HTTP request.

Unit 2: Lab 5

Related output:-



EXPECTED RESULTS

```
INFO[0000] Starting k6 run...
INFO[0000] The value of name field is:Mayank Narayana  source=console
INFO[0000] The value of name field is:mayank-leo@hotmail.com  source=console

running (00m00.2s), 0/1 VUs, 1 complete and 0 interrupted iterations
default ✓ [=====] 1 VUs  00m00.2s/10m0s  1/1 iters, 1 per VU

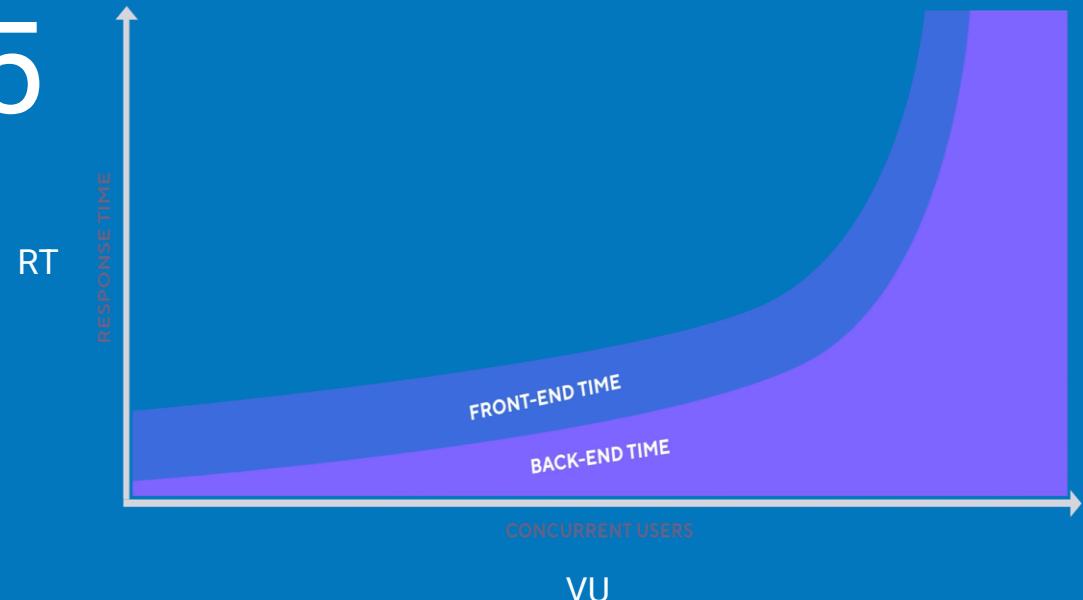
  ✓ name is correct
  ✓ email is correct

checks.....: 100.00% ✓ 2      ✘ 0
data_received.: 2.5 KB  10 KB/s
data_sent....: 109 B   458 B/s
http_req_blocked....: avg=97.15ms min=97.15ms med=97.15ms max=97.15ms p(90)=97.15ms p(95)=97.15ms
http_req_connecting....: avg=92.16ms min=92.16ms med=92.16ms max=92.16ms p(90)=92.16ms p(95)=92.16ms
http_req_duration....: avg=139.27ms min=139.27ms med=139.27ms max=139.27ms p(90)=139.27ms p(95)=139.27ms
  [ expected_response:true ]
http_req_failed....: avg=0.00ms min=0.00ms med=0.00ms max=0.00ms p(90)=0.00ms p(95)=0.00ms
http_req_receiving....: avg=164.31μs min=164.31μs med=164.31μs max=164.31μs p(90)=164.31μs p(95)=164.31μs
http_req_sending....: avg=73.58μs min=73.58μs med=73.58μs max=73.58μs p(90)=73.58μs p(95)=73.58μs
http_req_tls_handshaking....: avg=0s      min=0s      med=0s      max=0s      p(90)=0s      p(95)=0s
http_req_waiting....: avg=139.03ms min=139.03ms med=139.03ms max=139.03ms p(90)=139.03ms p(95)=139.03ms
http_reqs.....: 1      4.20453ms
iteration_duration....: avg=236.75ms min=236.75ms med=236.75ms max=236.75ms p(90)=236.75ms p(95)=236.75ms
iterations.....: 1      4.20452s
```

Unit 2: Lab 5



EXPECTED RESULTS



Unit 2: Lab 5

K6 performance and load testing

1. Install K6 on Host OS.
2. Create an K6 `script.js` that checks EXAMPLE.COM, including minimum 5 validations, 5 correlations (REGEX), 5 assertions, for testing as follows:
 1. Data Integrity,
 2. Performance,
 3. Minimal load (10 concurrent vus for 15 minutes).



Unit 2: EOF

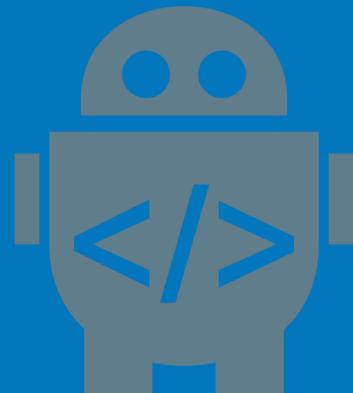


shaygi1@ac.sce.ac.il



Unit 3: Test Automation with AI

1. Introduction
2. Batch files (Windows)
3. Bash scripts (Linux)
4. Postman (Web)

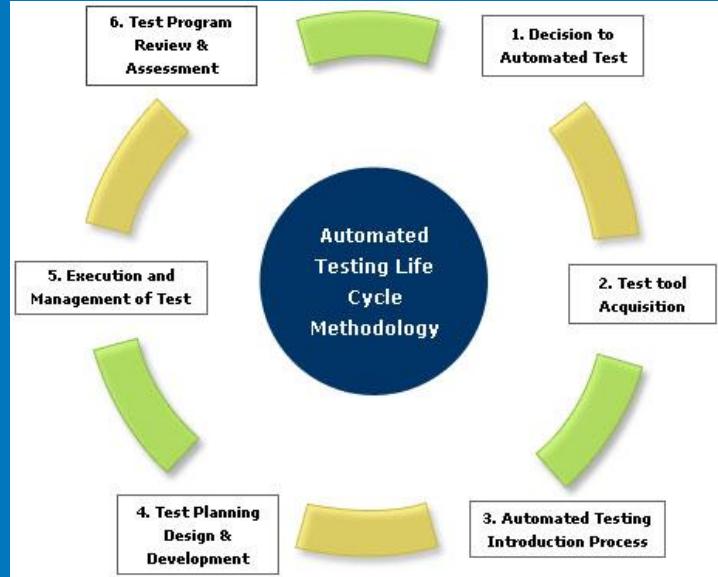


3.1 Introduction

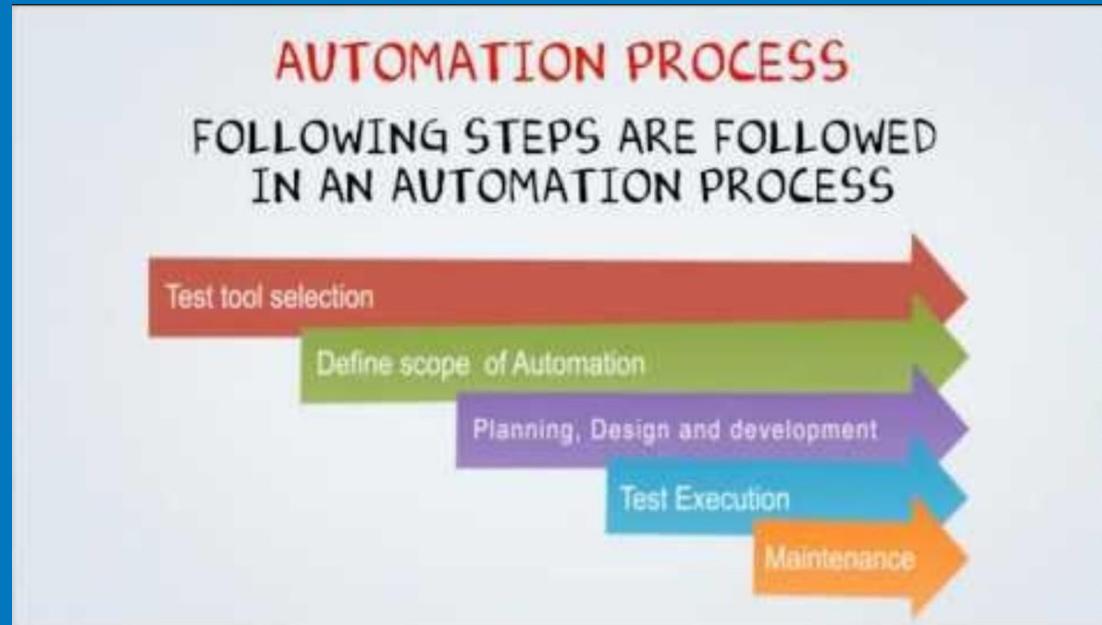
Test automation is a **process**, not an event.
Instruct the team to write **reusable test scripts**.
Otherwise **manual testing** is preferred.

Check the test tool prior to its introduction.
Finding out that it's not adequate halfway
through a project would be **devastating**.

Manage people's expectations of the tool.
If people expect too much, they will always be
disappointed - no matter what the outcome.



3.1 Introduction



3.1 Introduction

Bot is a software application that runs automated tasks (scripts) over the Internet. Typically, bots perform tasks that are both **simple and structurally repetitive**, at a **much higher rate** than would be possible for a human alone. The largest use of bots is in web spidering (web crawler), in which an **automated script fetches, analyzes and files information** from web servers at many times the speed of a human. More than half of all web traffic is made up of bots.



3.1 Introduction

Basic requirements from test automation tools:

1. Bot generation
2. Functional testing, regression testing
3. Data integrity, backward compatibility
4. Performance
5. Regular expressions
6. Validations, assertions



3.2 Batch files

A batch file is a **script file** in DOS, OS/2 and Microsoft Windows. It consists of a series of commands to be executed by the **command-line interpreter**, stored in a plain text file.

A batch file may contain any command the interpreter accepts **interactively** and use constructs that enable conditional branching and looping within the batch file, such as **IF**, **FOR**, and **GOTO** labels.



3.2 Batch files

Untitled - Notepad

File Edit Format View Help

```

ECHO - Displays text on the screen

@ECHO OFF - Hides the text that is normally output

START - Run a file with its default application

REM - Inserts a comment line in the program

MKDIR/RMDIR - Create and remove directories

DEL - Deletes a file or files

COPY - Copy a file or files

XCOPY - Allows you to copy files with extra options

FOR/IN/DO - Lets you specify files.

TITLE - Edit the title of the window.

```

will How to Write a Batch File



test.bat - Notepad

File Edit Format View Help

```

@echo off
title This is your first batch script!
echo Welcome to batch scripting!
pause

```

<https://www.c3scripts.com/tutorials/msdos/commands.html>

3.2 Batch files

What Is a Batch File?

A batch file is a file that contains one or more commands to be executed subsequently. Instead of typing those commands in Command Prompt or PowerShell, you create a file that runs those commands whenever the file is launched. This lets you accomplish the tasks of those commands.

A batch file uses ".bat" as its file extension. You run batch files the same way you run other files—simply double-click a file to launch it. No third-party apps are needed to open batch files.



3.2 Batch files

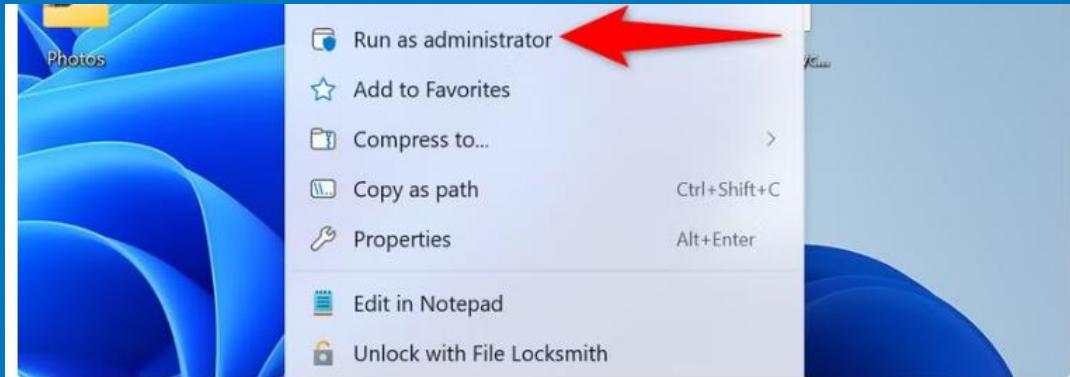
How to Create a Batch File

Creating a batch file is as easy as creating a plain text Notepad document. You create a new file in a [plain text editor](#), add the commands you want, and save the file as a batch file. Your system will run the file the same way regardless of how it was created as long as it's a proper batch file.

To create a batch file on Windows 11, open Windows Search (press Windows+S), type **Notepad**, and launch the app. Type the commands you want your file to have. The following sections have some useful commands that you can use to automate your tasks.



3.2 Batch files



If you always want to run a batch file as an admin (so you don't have to right-click the file and choose "Run as Administrator" each time), right-click the batch file and choose Show More Options > Send To > Desktop (Create Shortcut).

Right-click the newly created shortcut and select "Properties." Access the "Shortcut" tab, click "Advanced," enable "Run as Administrator," choose "OK," and select "Apply" followed by "OK."



3.2 Batch files

EXAMPLE

Emptying the Recycle Bin

You can make a batch file that automatically clears all the contents of the Recycle Bin. This frees up your storage space and declutters your machine.

To do that, use the following commands in a batch file.

Note

This batch file requires administrator privileges to run it.

```
@echo off
echo Emptying Recycle Bin for all drives...
powershell -Command "Clear-RecycleBin -Force -ErrorAction Ignore"
echo Recycle Bin emptied.
pause
```



3.2 Batch files

EXAMPLE

Clearing Temporary Files

Removing temporary files helps you free up your storage space and declutter your computer.

The following commands help you do that.

Note

Make sure to run this file as an admin to avoid running into any errors.

```
@echo off  
echo Clearing Temporary Files...  
del /q /f /s %temp%\*  
rd /s /q %temp%  
echo Temporary files cleared.  
pause
```



3.2 Batch files

EXAMPLE

Launching Multiple Apps at Once

If you often launch certain apps one after another, you can create a batch file that automatically launches all those apps for you. You can specify the apps to be launched in the commands.

```
@echo off
echo Launching apps...
start explorer
start chrome
start "" AppPath
echo All apps launched.
pause
```

3.2 Batch files

EXAMPLE

Back Up Files and Folders

To back up certain files and folders, you can make a batch file that automatically copies items from one source and pastes those items at another path.

```
@echo off
echo Backing up files...
xcopy "SourcePath" "DestinationPath" /e /i /h /y
echo Backup completed.
pause
```

3.2 Batch files

EXAMPLE

Back Up Files and Folders

To back up certain files and folders, you can make a batch file that automatically copies items from one source and pastes those items at another path.

```
@echo off
echo Backing up files...
xcopy "SourcePath" "DestinationPath" /e /i /h /y
echo Backup completed.
pause
```

3.2 Batch files

Reset Network

Often when you run into network problems, resetting your computer's IP address and flushing the DNS cache helps fix issues. You can use the following code in a batch file to perform these tasks.

Note

This batch file requires running as an administrator to function.

```
@echo off
echo Resetting network...
ipconfig /release
ipconfig /renew
ipconfig /flushdns
echo Network reset completed.
pause
```



Unit 3: Lab 6

/pattern/

Search with Regular Expression Batch

```
findstr /R pattern filename.txt
```

Examples - Search for the occurrence of all words ending with 'ing' in a file:

```
findstr /R [a-z]*ing filename.txt
```

```
findstr /R [A-Z]?[a-z]*ing filename.txt
```

Unit 3: Lab 6



Batch Scripting - cURL Bot

cURL is a command-line tool for getting or sending data including files using URL syntax.

Since cURL uses libcurl, it supports a range of common network protocols, currently including: HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, LDAP, IMAP, POP3, SMTP and more.

Unit 3: Lab 6



Get the main page from a web-server:

```
curl http://www.example.com/
```

Get a web page from a server using port 8000:

```
curl http://www.example.com:8000/
```

Get a file off an FTP server:

```
curl ftp://files.are.secure.com/secrets.txt
```

Get a web page and store in a local file with a specific name:

```
curl -o homepage.html http://www.example.com/
```

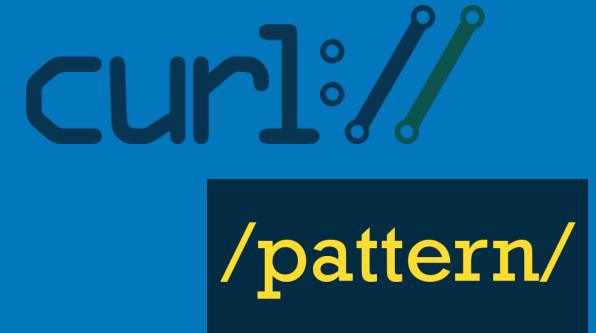
Get a file from an SSH server using SFTP:

```
curl -u username sftp://example.com/etc/issue
```

Unit 3: Lab 6

Create a batch file bot for:

1. Extracting the date from the web page: www.timeanddate.com
2. Extracting the IP address from the web page: www.whatismyip.com
3. Extracting the main headline from the web page: www.ynet.co.il
4. Extracting the download link from: www.bugzilla.org



Unit 3: Lab 6

Create a batch file bot for:

5. Extracting the first 'Tutorial' name from: JMeter.apache.org
6. Extracting the names of all NFT course students from: moodle.sce.ac.il
7. Extracting 'NFT jobs' from the web page: www.linkedin.com
8. Extracting 'software non functional testing' books from: www.amazon.com



/pattern/

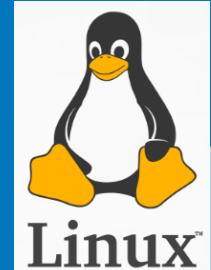
3.3 Bash scripts

Bash is a Unix shell and command language.

Bash is a command processor that typically runs in a text window where the user types commands that cause actions.

Bash can also read and execute commands from a file, called a shell script. Like all Unix shells, it supports wildcard matching, piping, variables, and control structures for condition-testing and iteration.

```
1  #!/bin/bash
2  #INPUT_SAMPLE_LIST=$1
3  cd /Volumes/PhilDrive_EMS/TestDec7/snvs_postprocess/
4  ;;
5  . paths.txt
6  ;;
7
8
9
10
11 echo "Debug level set for $DEBUG_LEVEL"
12 echo "log found in scripts directory"
13 ;;
14
15 cp $HIGH_SNPs_OUT ./;
16 cp $LOW_SNPs_OUT ./;
17 cp $GERM_SNPs_OUT ./;
18 # echo "${SCRIPT_DIR}/run_somatic_mutation_analysis ${i} no_f
19 if [ $DEBUG_LEVEL -gt 0 ]
20 then
21   echo "INFO: ${SCRIPT_DIR}run_somatic_mutation_analysis.sh $SAM
22 `basename ${LOW_SNPs_OUT}` `basename ${GERM_SNPs_OUT}` `basename
23 ${D_BAM_FILE}` ${G_BAM_FILE}\n">>${LOG}
24
25
```



3.3 Bash scripts

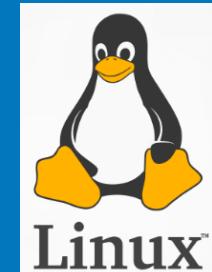
```
ubuntu@ubuntu-VirtualBox:~$ echo "Hello World"
Hello World
ubuntu@ubuntu-VirtualBox:~$
```

```
GNU nano 2.8.6                               File: First.sh                         Modified

#!/bin/bash
echo "Hello World"

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File   ^\ Replace   ^U Uncut Text^T To Linter ^L Go To Line
```

```
ubuntu@ubuntu-VirtualBox:~$ bash First.sh
Hello World
ubuntu@ubuntu-VirtualBox:~$ chmod a+x First.sh
ubuntu@ubuntu-VirtualBox:~$ ./First.sh
Hello World
ubuntu@ubuntu-VirtualBox:~$
```

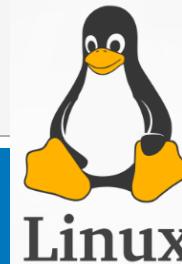


3.3 Bash scripts

for.sh

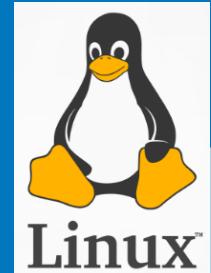
```
#!/bin/sh
for i in 1 2 3 4 5
do
    echo "Looping ... number $i"
done
```

```
Looping .... number 1
Looping .... number 2
Looping .... number 3
Looping .... number 4
Looping .... number 5
```



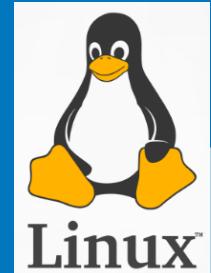
3.3 Bash scripts

```
while.sh
#!/bin/sh
INPUT_STRING=hello
while [ "$INPUT_STRING" != "bye" ]
do
    echo "Please type something in (bye to quit)"
    read INPUT_STRING
    echo "You typed: $INPUT_STRING"
done
```



3.3 Bash scripts

```
if [ something ]; then
    echo "Something"
elif [ something_else ]; then
    echo "Something else"
else
    echo "None of the above"
fi
```



3.3 Bash scripts

```
talk.sh
#!/bin/sh

echo "Please talk to me ..."
while :
do
    read INPUT_STRING
    case $INPUT_STRING in
        hello)
            echo "Hello yourself!"
            ;;
        bye)
            echo "See you again!"
            break
            ;;
        *)
            echo "Sorry, I don't understand"
            ;;
    esac
done
echo
echo "That's all folks!"
```



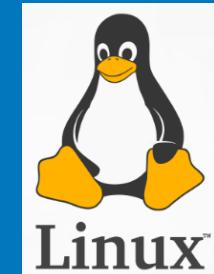
3.3 Bash scripts

Command	Description	Example
&	Run the previous command in the background	ls &
&&	Logical AND	if ["\$foo" -ge "0"] && ["\$foo" -le "9"]
	Logical OR	if ["\$foo" -lt "0"] ["\$foo" -gt "9"]
^	Start of line	grep "^foo"
\$	End of line	grep "foo\$"
=	String equality (cf. -eq)	if ["\$foo" = "bar"]
!	Logical NOT	if ["\$foo" != "bar"]
\$\$	PID of current shell	echo "my PID = \$\$"
\$!	PID of last background command	ls & echo "PID of ls = \$!"
\$?	exit status of last command	ls ; echo "ls returned code \$?"
\$0	Name of current command (as called)	echo "I am \$0"
\$1	Name of current command's first parameter	echo "My first argument is \$1"
\$9	Name of current command's ninth parameter	echo "My ninth argument is \$9"
\$@	All of current command's parameters (preserving whitespace and quoting)	echo "My arguments are \$@"
\$*	All of current command's parameters (not preserving whitespace and quoting)	echo "My arguments are \$*"



3.3 Bash scripts

-eq	Numeric Equality	if ["\$foo" -eq "9"]
-ne	Numeric Inquality	if ["\$foo" -ne "9"]
-lt	Less Than	if ["\$foo" -lt "9"]
-le	Less Than or Equal	if ["\$foo" -le "9"]
-gt	Greater Than	if ["\$foo" -gt "9"]
-ge	Greater Than or Equal	if ["\$foo" -ge "9"]
-z	String is zero length	if [-z "\$foo"]
-n	String is not zero length	if [-n "\$foo"]
-nt	Newer Than	if ["\$file1" -nt "\$file2"]
-d	Is a Directory	if [-d /bin]
-f	Is a File	if [-f /bin/ls]
-r	Is a readable file	if [-r /bin/ls]
-w	Is a writable file	if [-w /bin/ls]
-x	Is an executable file	if [-x /bin/ls]
(...)	Function definition	function myfunc() { echo hello }



3.3 Bash scripts

Bash has a built-in regular expression comparison operator, represented by `=~`

```
if [[ $digit =~ [0-9] ]]; then
    echo "$digit is a digit"
else
    echo "oops"
fi
```

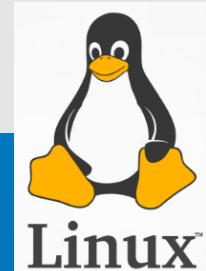
```
echo -n "Your answer> "
read REPLY
if [[ $REPLY =~ ^[0-9]+$ ]]; then
    echo Numeric
else
    echo Non-numeric
fi
```

```
#!/bin/bash

read -p "Enter email: " email

if [[ "$email" =~ ^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$ ]]
then
    echo "This email address looks fine: $email"
else
    echo "This email address is flawed: $email"
fi
```

`^` asserts position at start of a line
`$` asserts position at the end of a line



3.3 Bash scripts

Bash has a built-in regular expression comparison operator, represented by `=~`

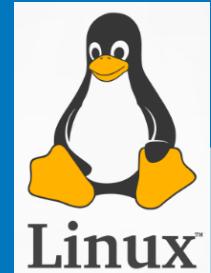
```
#!/bin/bash

if [ $# != 1 ]; then
    echo "Usage: $0 address"
    exit 1
else
    ip=$1
fi

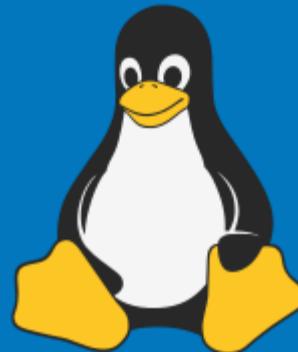
if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
    echo "Looks like an IPv4 IP address"
elif [[ $ip =~ ^[A-Fa-f0-9:]+$ ]]; then
    echo "Could be an IPv6 IP address"
else
    echo "oops"
fi
```



conditional check used to evaluate the number of arguments passed to the script.



Unit 3: Lab 7



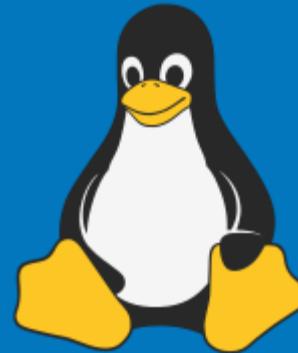
curl://

/pattern/

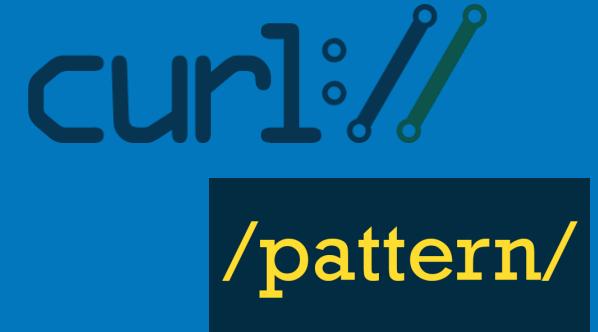
Create a bash file bot for:

1. Extracting the date from the web page: www.timeanddate.com
2. Extracting the IP address from the web page: www.whatismyip.com
3. Extracting the main headline from the web page: www.ynet.co.il
4. Extracting the download link from: www.bugzilla.org

Unit 3: Lab 7



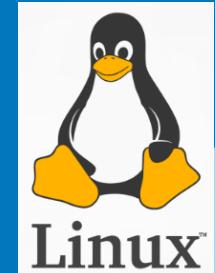
Create a bash file bot for:



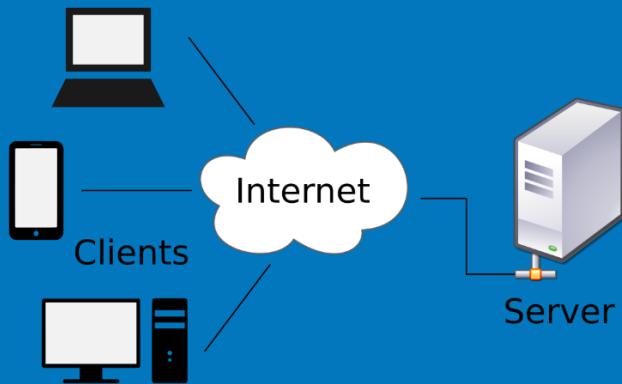
5. Extracting the first 'Tutorial' name from: JMeter.apache.org
6. Extracting the names of all NFT course students from: moodle.sce.ac.il
7. Extracting 'NFT jobs' from the web page: www.linkedin.com
8. Extracting 'software non functional testing' books from: www.amazon.com

3.3 Bash scripts for NFT

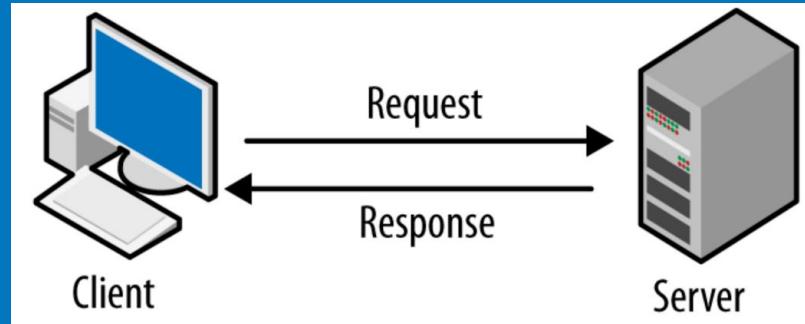
1. Create an python app that runs on Linux and estimates when would the local hard drive be full. (Free space = 0 MB)
2. Check the performance of that app, including:
Response time, CPU usage, RAM usage, Disk usage.
3. Program a BASH file that runs that app in a loop for 24 hours and monitors the app's performance. Assert on memory leaks and other performance issues in real-time.



3.4 Postman



- GUI
- Usability
- Accessibility

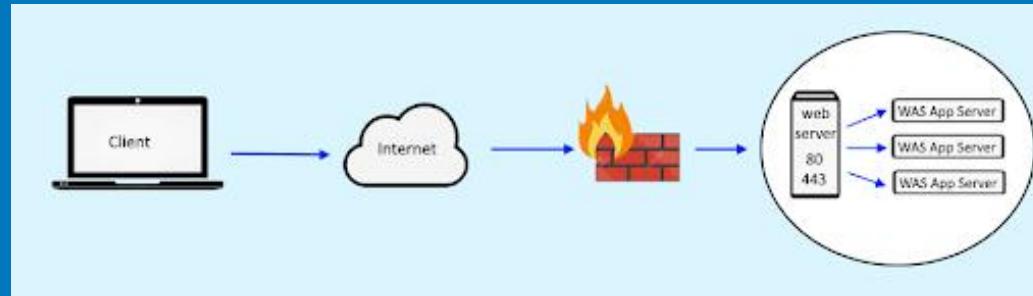


- Client-Server Model
- Protocol
- API

3.4 Postman



Much of the Internet is based on the client-server model. In this model, user devices communicate via a network with centrally located servers to get the data they need, instead of communicating with each other. End user devices such as laptops, smartphones, and desktop computers are 'clients' of the servers, as if they were customers obtaining services from a company. Client devices send requests to the servers for webpages or applications, and the servers serve up responses.

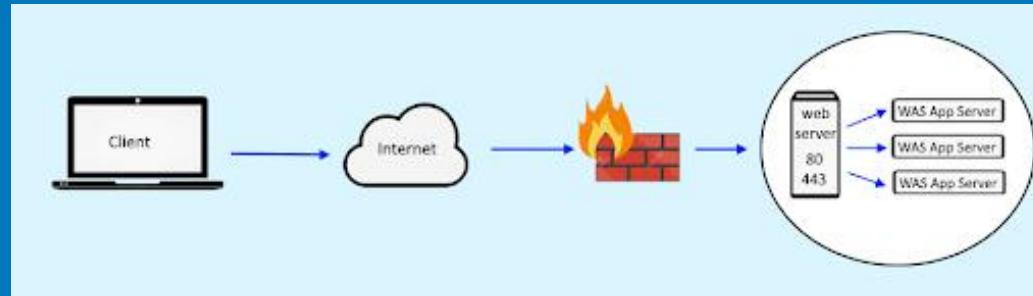


3.4 Postman



POSTMAN

The client-server model is used because servers are typically more powerful and more reliable than user devices. They also are constantly maintained and kept in controlled environments to make sure they're always on and available; although individual servers may go down, there are usually other servers backing them up. Meanwhile, users can turn their devices on and off, or lose or break their devices, and it should not impact Internet service for other users.



3.4 Postman

In web development, 'client side' refers to everything in a web application that is displayed or takes place on the client (end user device). This includes what the user sees, such as text, images, and the rest of the UI, along with any actions that an application performs within the user's browser.

Markup languages like HTML and CSS are interpreted by the browser on the client side. In addition, many contemporary developers are including client-side processes in their application architecture and moving away from doing everything on the server side; business logic for dynamic webpages, for instance, usually runs client side in a modern web application. Client-side processes are almost always written in JavaScript.



The client side is also known as the frontend, although these two terms do not mean precisely the same thing. Client-side refers solely to the location where processes run, while frontend refers to the kinds of processes that run client-side.



3.4 Postman

Much like with client side, 'server side' means everything that happens on the server, instead of on the client. In the past, nearly all business logic ran on the server side, and this included rendering dynamic webpages, interacting with databases, identity authentication, and push notifications.

The problem with hosting all these processes on the server side is that each request involving one of them must travel all the way from the client to the server, every time. This introduces a great deal of latency. For this reason, contemporary applications run more code on the client side; one use case is rendering dynamic webpages in real time by running scripts within the browser that make changes to the content a user sees.

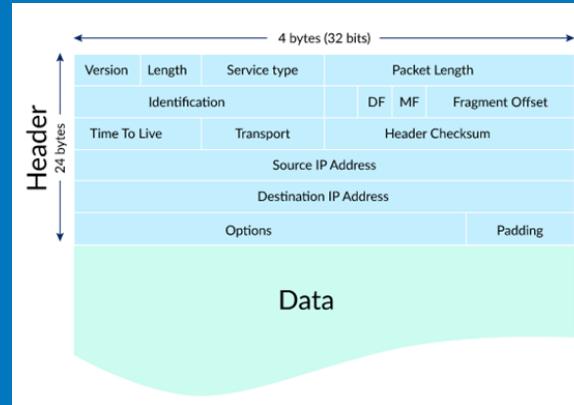
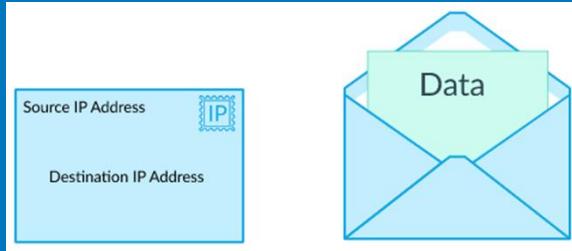


Like with 'frontend' and 'client-side,' backend is also a term for the processes that take place on the server, although backend only refers to the types of processes and server-side refers to the location where processes run.



POSTMAN

3.4 Postman



The Internet Protocol (IP) is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. Its routing function enables internetworking, and essentially establishes the Internet.

IP has the task of delivering packets from the source host to the destination host solely based on the IP addresses in the packet headers. For this purpose, IP defines packet structures that encapsulate the data to be delivered. It also defines addressing methods that are used to label the datagram with source and destination information.

3.4 Postman



When a client wants to communicate with a server, either the final server or an intermediate proxy, it performs the following steps:

1. Open a TCP connection: The TCP connection is used to send a request, or several, and receive an answer. The client may open a new connection, reuse an existing connection, or open several TCP connections to the servers.
2. Send an HTTP message: HTTP messages (before HTTP/2) are human-readable. With HTTP/2, these simple messages are encapsulated in frames, making them impossible to read directly, but the principle remains the same. For example:

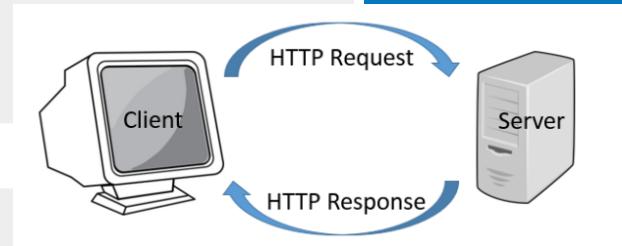
```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

3. Read the response sent by the server, such as:

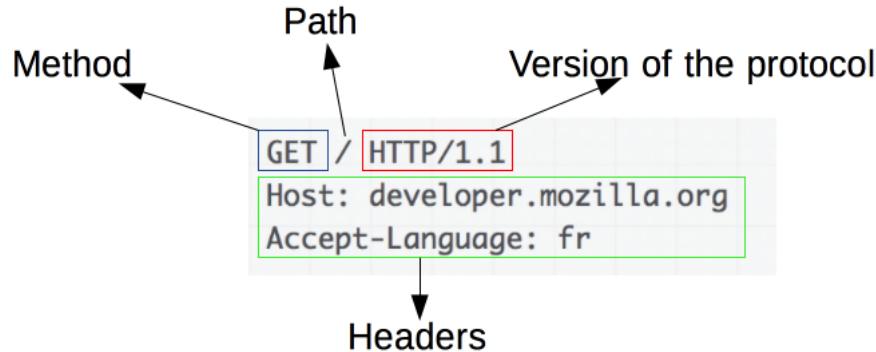
```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html
```

```
<!DOCTYPE html... (here comes the 29769 bytes of the requested web page)
```

4. Close or reuse the connection for further requests.



3.4 Postman



Requests consists of the following elements:

- An HTTP method, usually a verb like `GET`, `POST` or a noun like `OPTIONS` or `HEAD` that defines the operation the client wants to perform. Typically, a client wants to fetch a resource (using `GET`) or post the value of an HTML form (using `POST`), though more operations may be needed in other cases.
- The path of the resource to fetch; the URL of the resource stripped from elements that are obvious from the context, for example without the protocol (`http://`), the domain (here, `developer.mozilla.org`), or the TCP port (here, `80`).
- The version of the HTTP protocol.
- Optional headers that convey additional information for the servers.
- Or a body, for some methods like `POST`, similar to those in responses, which contain the resource sent.

3.4 Postman



HTTP Methods and Their Meaning

Method	Meaning
GET	Read data
POST	Insert data
PUT or PATCH	Update data, or insert if a new id
DELETE	Delete data

3.4 Postman

HTTP Method	Request has body	Response has body	Safe	Idempotent	Cachable
GET	NO	YES	YES	YES	YES
POST	YES	YES	NO	NO	YES
PUT	YES	YES	NO	YES	NO
DELETE	YES	YES	NO	YES	NO
TRACE	NO	YES	YES	YES	NO
OPTIONS	NO	YES	YES	YES	NO
CONNECT	NO	YES	NO	NO	NO
PATCH	YES	YES	NO	NO	NO



An HTTP method is **idempotent** if an identical request can be made once or several times in a row with the same effect while leaving the server in the same state. In other words, an idempotent method should not have any side-effects (except for keeping statistics).

3.4 Postman



Which One to Prefer?

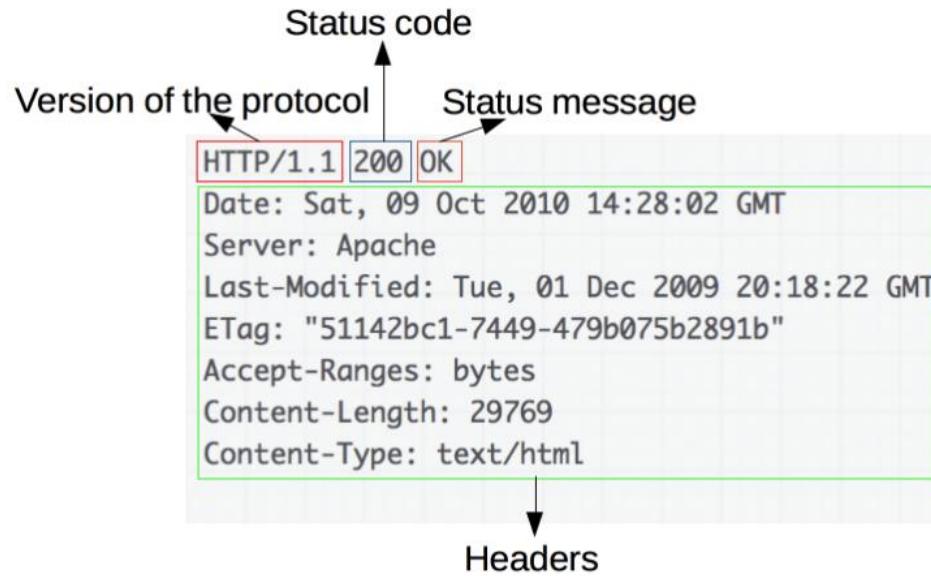
- **For updates:** Prefer **PUT** or **PATCH** over **POST**.
- **For partial updates:** **PATCH** is the best choice.
- **For full replacements:** Use **PUT**.
- **For creating new resources:** Use **POST**.

If your goal is updating resources efficiently:

- **PATCH** (partial updates) is generally preferred over **POST** (creation).
- **PUT** can be better when the entire resource is known and should be replaced.

HTTP Method	Request has body	Response has body	Safe	Idempotent	Cachable
GET	NO	YES	YES	YES	YES
POST	YES	YES	NO	NO	YES
PUT	YES	YES	NO	YES	NO
DELETE	YES	YES	NO	YES	NO
TRACE	NO	YES	YES	YES	NO
OPTIONS	NO	YES	YES	YES	NO
CONNECT	NO	YES	NO	NO	NO
PATCH	YES	YES	NO	NO	NO

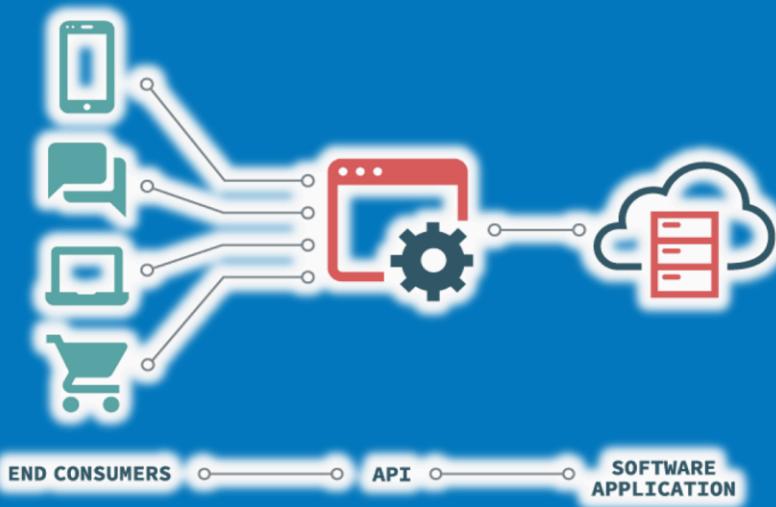
3.4 Postman



Responses consist of the following elements:

- The version of the HTTP protocol they follow.
- A status code, indicating if the request was successful, or not, and why.
- A status message, a non-authoritative short description of the status code.
- HTTP headers, like those for requests.
- Optionally, a body containing the fetched resource.

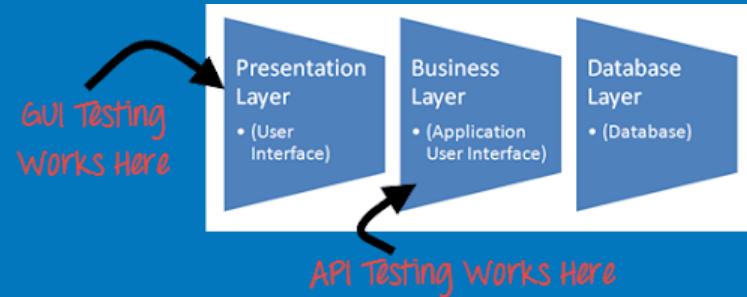
3.4 Postman



Application Programming Interface (API) is an interface that defines interactions between multiple software applications or mixed hardware-software intermediaries. It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc.

Through information hiding, APIs enable modular programming, allowing users to use the interface independently of the implementation.

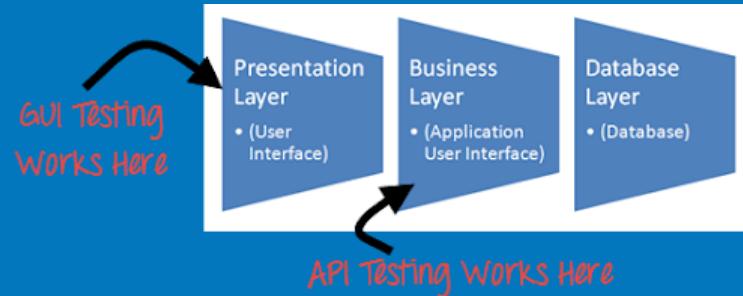
3.4 Postman



Application Programming Interface (API) Testing is a type of software testing that involves testing application programming interfaces (APIs) directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance, and security.

Since APIs lack a GUI, API testing is performed at the message layer. API testing is now considered critical for automating testing because APIs now serve as the primary interface to application logic and because GUI tests are difficult to maintain with the short release cycles and frequent changes commonly used with Agile software development.

3.4 Postman



1. **GUI layer testing use cases:**

Manual functional testing, Regression testing, Usability testing, Accessibility testing.

GUI layer can also be used for automation; however, it requires exhaustive maintenance efforts.

2. **Business layer (API) testing use cases:**

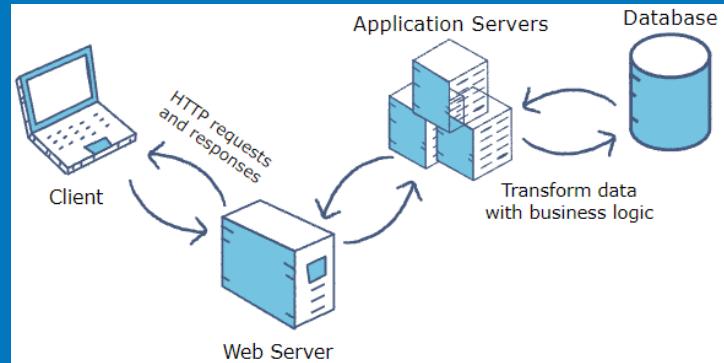
Testing Automation, Application scalability testing, Application Performance and Load testing.

3. **Database layer testing use cases:**

SQL command testing, Data integrity and correctness testing, Database scalability testing, Database Performance and Load testing.

3.4 Postman

- A web server accepts and fulfills requests from clients for static content (i.e., HTML pages, files, images, and videos) from a website. Web servers handle HTTP requests and responses only.
- An application server exposes business logic to the clients, which generates dynamic content. It is a software framework that transforms data to provide the specialized functionality offered by a business, service, or application. Application servers enhance the interactive parts of a website that can appear differently depending on the context of the request.



Web Server and Web Application Server

Unit 3: Test Automation

3.4 Postman

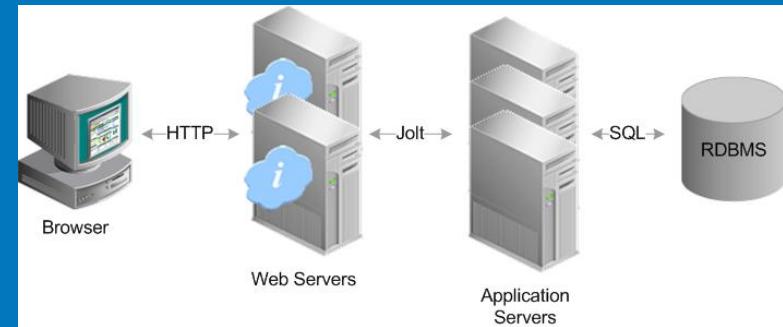


Web Server

- Deliver static content.
- Content is delivered using the HTTP protocol only.
- Serves only web-based applications.
- No support for multi-threading.
- Facilitates web traffic that is not very resource intensive.

Application Server

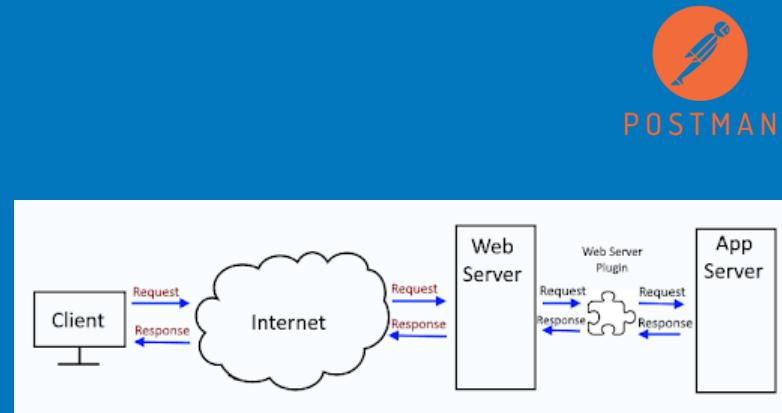
- Delivers dynamic content.
- Provides business logic to application programs using several protocols (including HTTP).
- Can serve web and enterprise-based applications.
- Uses multi-threading to support multiple requests in parallel.
- Facilitates longer running processes that are very resource-intensive.



3.4 Postman

Key differences in features of
Web Server and Application Server:

1. Web Server is designed to serve HTTP Content. App Server can also serve HTTP Content but is not limited to just HTTP. It can be provided other protocol support such as RMI/RPC.
2. Web Server is mostly designed to serve static content, though most Web Servers have plugins to support scripting languages like Perl, PHP, ASP, JSP etc. through which these servers can generate dynamic HTTP content.

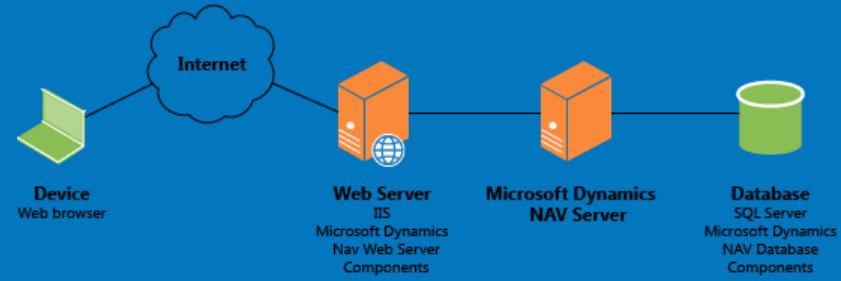


3.4 Postman



Key differences in features of
Web Server and Application Server:

3. Most of the application servers have Web Server as integral part of them, that means App Server can do whatever Web Server is capable of. Additionally, App Server have components and features to support Application-level services such as Connection Pooling, Object Pooling, Transaction Support, Messaging services etc.

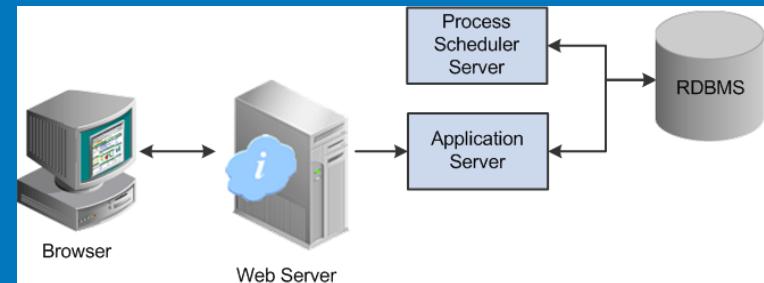


3.4 Postman



Key differences in features of
Web Server and Application Server:

4. As web servers are well suited for static content and app servers for dynamic content, most of the production environments have web server acting as reverse proxy to app server. That means while servicing a page request, static contents (such as images/Static HTML) are served by web server that interprets the request. Using filtering technique (mostly extension of requested resource) web server identifies dynamic content request and transparently forwards to app server.

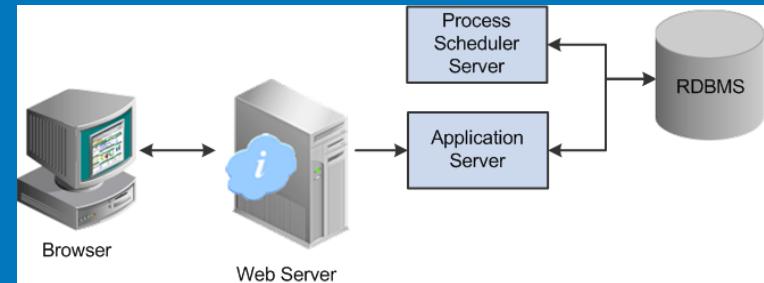


3.4 Postman



Key differences in features of
Web Server and Application Server:

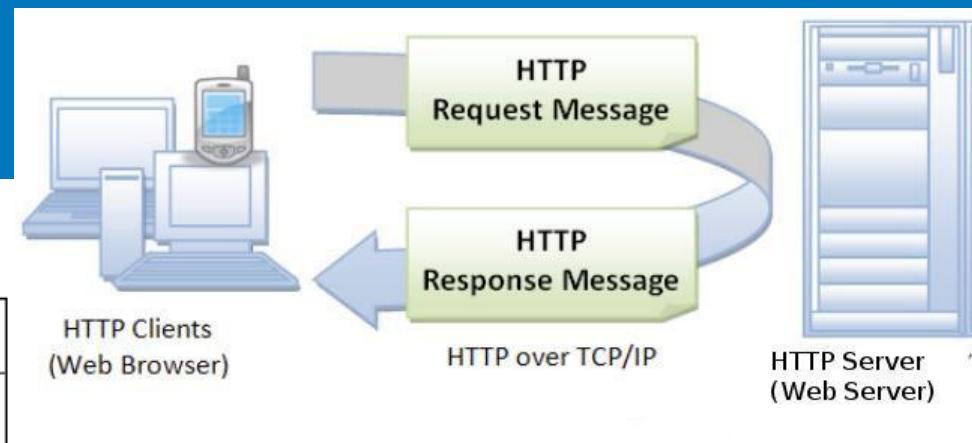
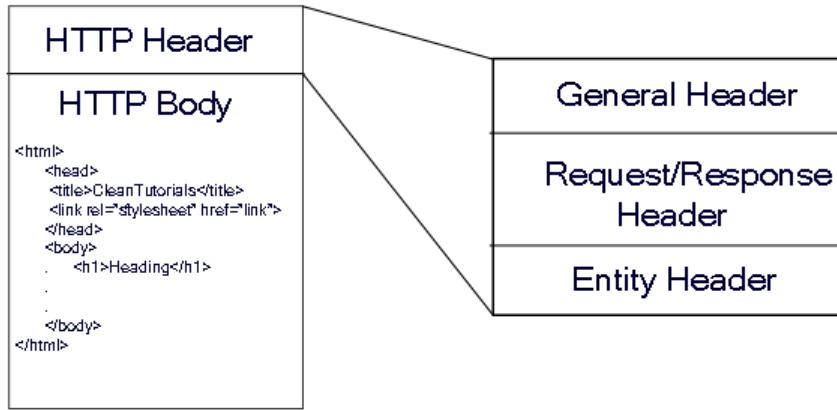
4. As web servers are well suited for static content and app servers for dynamic content, most of the production environments have web server acting as reverse proxy to app server. That means while servicing a page request, static contents (such as images/Static HTML) are served by web server that interprets the request. Using filtering technique (mostly extension of requested resource) web server identifies dynamic content request and transparently forwards to app server.



3.4 Postman



HTTP Request/response



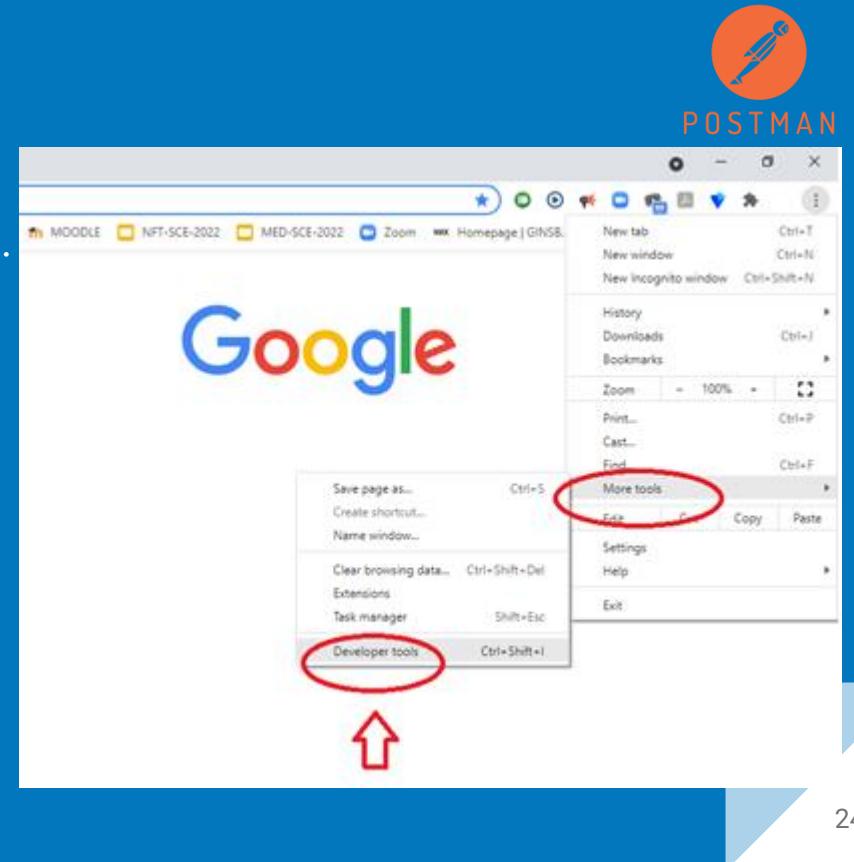
3.4 Postman

F12 developer tools is a suite of tools to help you build and debug and test web pages applications.

Creating web pages and applications requires coding and testing, as well as the right tools to find and debug issues that will inevitably pop up.

Current web browsers provide a view of your rendered code, and F12 tools provide a view of how those pages are interpreted on a code level by the web browser.

F12 tools also help identify and report about elements on the page, such as links, and image reports.



3.4 Postman

What is Postman?

Postman is an [API platform](#) for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.



3.4 Postman



- Postman is a popular API client for testing APIs by allowing users to create and save simple and complex HTTP requests as well as read their responses.
- Postman offers a simple user-friendly interface.
- Postman allows creating individual JavaScript test scripts.

Download Postman from: <https://www.postman.com/downloads/>

3.4 Postman

The screenshot shows the Postman application window. At the top, there are two tabs: "POST https://api.dotcom-monit..." and "POST https://api.dotcom-monit... X". The second tab is selected. Below the tabs, the URL "https://api.dotcom-monitor.com/config_api_v1/login" is entered. The "Headers (10)" tab is selected in the navigation bar. Under "Headers (1)", there is one entry: "Content-Type" with the value "application/json". In the bottom panel, the response status is "200 OK", time is "816ms", and size is "1017 B". The response body is displayed as JSON:

```
1  {
2      "Success": true,
3      "Result": "OK"
4 }
```



3.4 Postman



Body Cookies (1) Headers (7) Test Results Security

Pretty Raw Preview Visualize JSON

```
1 {  
2 >   "args": { ...  
5   },  
6 >   "headers": { ...  
17  },  
18   "url": "https://postman-echo.com/get?id=123&ty"  
19 }
```

Body Cookies Headers (21) Test Results

Pretty Raw Preview Visualize

```
{"type": "FeatureCollection", "metadata": {"gov/fdsnws/event/1/query?format=geojson&event\_type=Earthquakes", "status": 200, "api": "1.13.1",
```

3.4 Postman

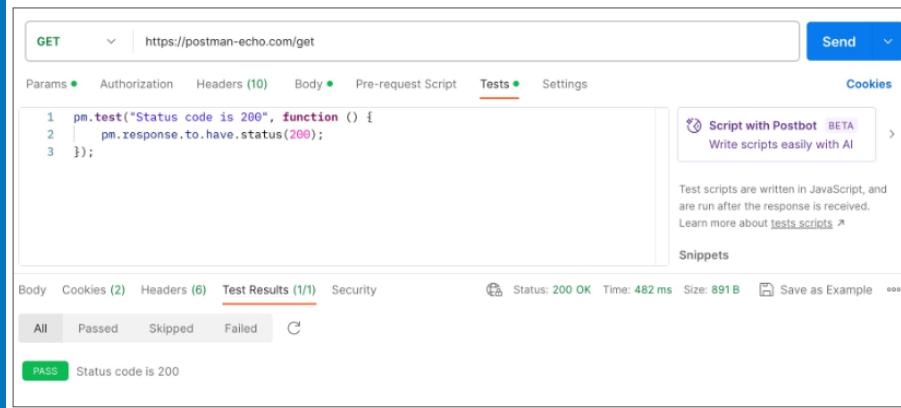


To write your first test script, open a request in Postman, then select the **Tests** tab. Enter the following JavaScript code:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

This code uses the `pm` library to run the `test` method. The text string will appear in the test output. The function inside the test represents an assertion. Postman tests can use [Chai Assertion Library BDD](#) syntax, which provides options to optimize how readable your tests are to you and your collaborators. In this case, the code uses BDD chains `to.have` to express the assertion.

This test checks the response code returned by the API. If the response code is `200`, the test will pass, otherwise it will fail. Select **Send** and go to the **Test Results** tab in the response area.



The screenshot shows the Postman interface for a GET request to `https://postman-echo.com/get`. The **Tests** tab is selected, displaying the following JavaScript code:

```
1 pm.test("Status code is 200", function () {
2     pm.response.to.have.status(200);
3});
```

The **Test Results** tab shows a single test result: **PASS** Status code is 200. The status bar at the bottom indicates **Status: 200 OK**, **Time: 482 ms**, and **Size: 891 B**.



3.4 Postman



<https://blog.postman.com/postman-api-performance-testing/>

**Test your API's performance by
simulating real-world traffic with
Postman**



Malvika Chaudhary

June 15, 2023 · 8 mins

3.4 Postman



<https://learning.postman.com/docs/introduction/overview/>

Overview

Welcome to the Postman Learning Center docs! This is the place to find official information on how to use Postman in your API projects.

If you're learning to carry out a specific task or workflow in Postman, check out the following topics to find resources:

Getting started

To get started using Postman, check out the [get started](#) section.

Sending requests

You can send requests in Postman to connect to APIs you are working with. To learn more about how to send requests, see [sending requests](#) and learn how to [send your first request](#).

Writing scripts

Postman has a powerful runtime based on Node.js that enables you to add dynamic behavior to requests and collections. You can write scripts that run before or after request execution to perform API tests, build requests that can contain dynamic parameters, pass data between requests, and more. To learn more about scripts, see [scripting in Postman](#).

3.4 Postman

Exercise



EXAMPLE

Send the following GET request:

<https://restcountries.eu/rest/v2/alpha/nor>

Use JavaScript tests to verify the following in the response:

1. Verify that the returned response code was 200 OK.
2. Verify that the Content-Type contains 'application/json'.
3. Verify that the country name is 'Norway'.



3.4 Postman

Solution



EXAMPLE

GET https://restcountries.eu/rest/v2/alpha/nor

Params Authorization Headers (7) Body Pre-request Script **Tests** ● Settings Cookies Code

```
1 pm.test("is 200OK", function(){
2     pm.response.to.have.status(200);
3 })
4
5 pm.test("verify content type is json", function(){
6     pm.response.to.have.header("Content-Type");
7     pm.expect(pm.response.headers.get("Content-Type")).to.contain("application/json");
8 })
9
10 pm.test("Verify joke returned", function(){
11     const responseJson = pm.response.json();
12     pm.expect(responseJson.name).eq1("Norway");
13 })
```

Test scripts are written in JavaScript, and are run after the response is received.
[Learn more about tests scripts](#)

SNIPPETS

[Get an environment variable](#)
[Get a global variable](#)
[Get a variable](#)
[Set an environment variable](#)
[Set a global variable](#)
[Clear an environment variable](#)

Body Cookies (1) Headers (16) **Test Results (3/3)**

All Passed Skipped Failed

PASS is 200OK
PASS verify content type is json
PASS Verify joke returned

Status: 200 OK Time: 534 ms Size: 1.94 KB Save Response ▾



3.4 Postman

.[RegEx]*



<https://community.postman.com/t/extract-string-from-response/7473>

Extract string from response

Help variables environments



Lukas.lobnig

1 Aug '19

My response looks like this

```
{  
  "code": 201,  
  "message": "192.168.1.106/repo/api/v1/report/23",  
  "type": "INFO"  
}
```

But I only need the 23 as environment variable.

My test script looks like this what do I have to do to only get the 23?

```
pm.environment.set('DownloadLink', pm.response.json().message);
```

3.4 Postman

.[RegEx]*



<https://community.postman.com/t/regex-to-assert-html-in-a-response-body/33969>

RegEx to assert HTML in a response body

Help ■ html ■ regex

w4dd325 Mar '22

I have this line return within HTML in a response body;
`<script type="text/javascript" src="/main.fed3fbceb96763d430c.chunk.js"></script>`

The string is a hash of letters and numbers and I'm trying to assert that the value in "src" is
`/main.REGEX.chunk.js`

I have been reading other posts and the best I have come up with is this;

```
const response = pm.response.text();
let my_regex = new RegExp(pm.globals.get('regEx'));
pm.test("Body matches string", function () {
    pm.expect(response).to.include(`src="/main.${my_regex}.chunk.js"`);
});
```

However, it still doesn't work, can anyone help me alter this to work properly?
Any guidance would be appreciated.

3.4 Postman

Scenario: Validate an email format in a JSON response.

JSON Response:

```
{  
  "user": {  
    "email": "test@example.com"  
  }  
}
```

Test Script:

```
pm.test("Email format is valid", () => {  
  const email = pm.response.json().user.email;  
  const emailRegex = /^[^s@]+@[^s@]+\.[^s@]+$/;  
  pm.expect(email).to.match(emailRegex);  
});
```

This checks if the email field matches a valid email format.

.[RegEx]*

EXAMPLE



Unit 3: Lab 8



Create a Postman bot for:

1. Extracting the date from the web page: www.timeanddate.com
2. Extracting the IP address from the web page: www.whatismyip.com
3. Extracting the main headline from the web page: www.ynet.co.il
4. Extracting the download link from: www.bugzilla.org

Unit 3: Lab 8



Create a Postman bot for:

5. Extracting the first 'Tutorial' name from: JMeter.apache.org
6. Extracting the names of all NFT course students from: moodle.sce.ac.il
7. Extracting 'NFT jobs' from the web page: www.linkedin.com
8. Extracting 'software non functional testing' books from: www.amazon.com

Unit 3: EOF



shaygi1@ac.sce.ac.il

Unit 4: Software Usability

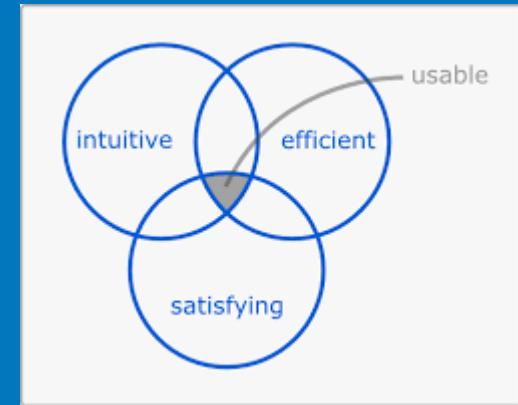
1. Concept
2. Testability



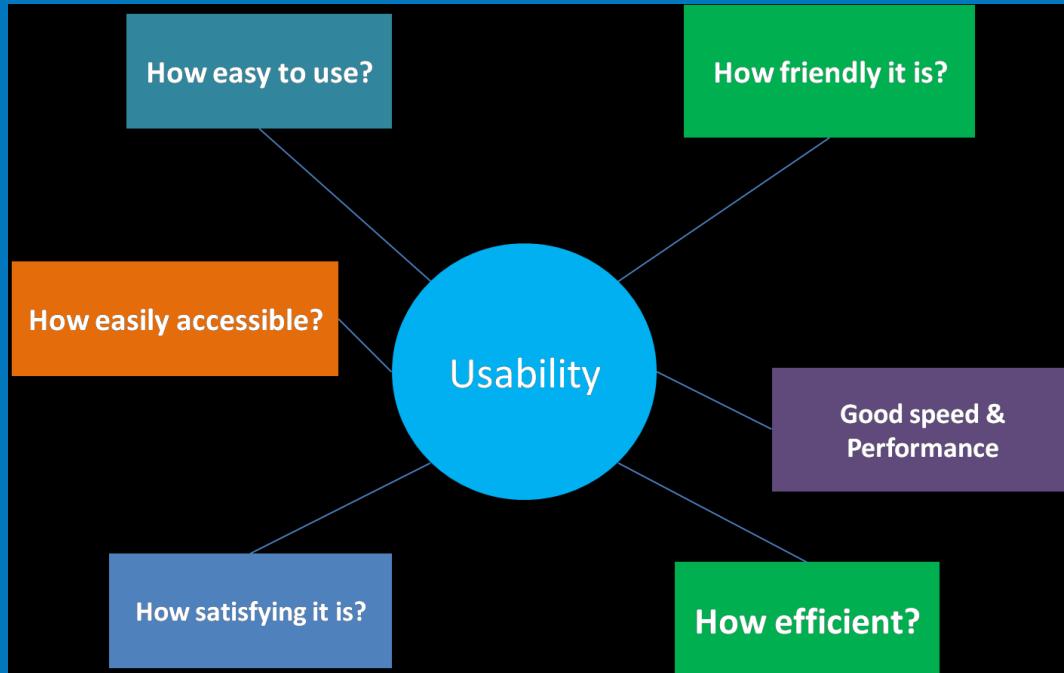
4.1 Software Usability Concept

Usability is the degree to which a software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use.

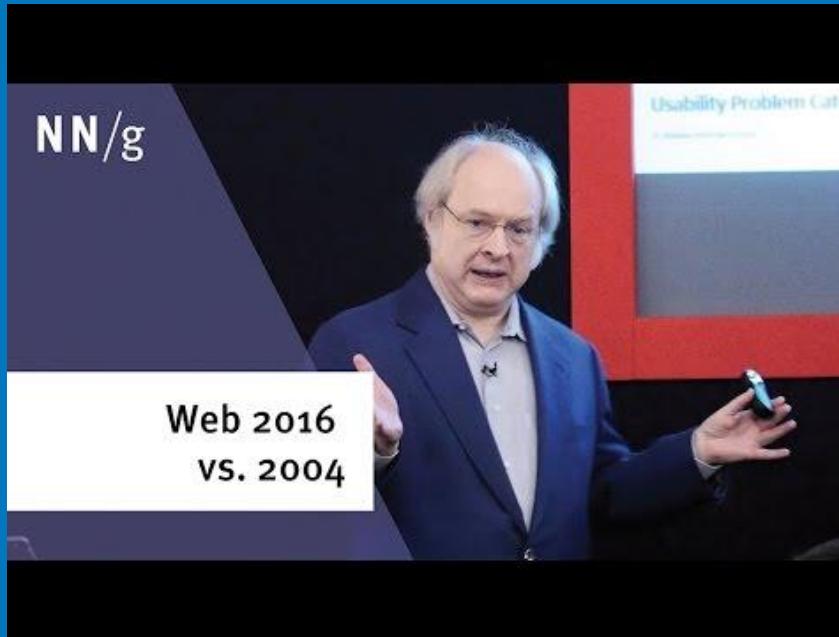
The object of use can be a software application, website, mobile application, or anything a human interacts with.



4.2 Software Usability Testability



4.2 Software Usability Testability



4.2 Software Usability Testability



Unit 4: Lab 9

1. Mobile Applications - Document Scanning
2. Mobile Applications - Ringtone Editing
3. Mobile Applications - Photo Editing



Task:

Select a mobile application and implement a quantitative usability testing method on it.

Unit 4: Lab 10

1. Desktop Applications - Document Scanning
2. Desktop Applications - Ringtone Editing
3. Desktop Applications - Photo Editing



Task:

Select a desktop application and implement a quantitative usability testing method on it.

Unit 4: EOF



shaygi1@ac.sce.ac.il

Unit 5: Performance & Load Testing

1. Introduction
2. Black Friday & Cyber Monday
3. Apache JMeter



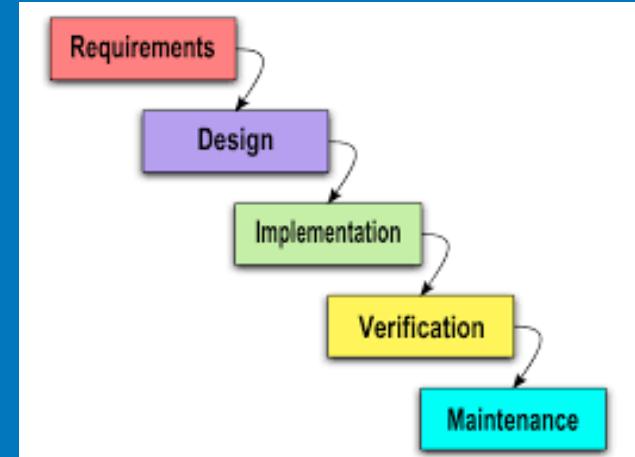
5.1 Introduction:

- Non-Functional Testing - concept based testing:
 - Load/Performance testing
 - Compatibility testing
 - Localization testing
 - Security testing
 - Reliability testing
 - Stress testing
 - Usability testing

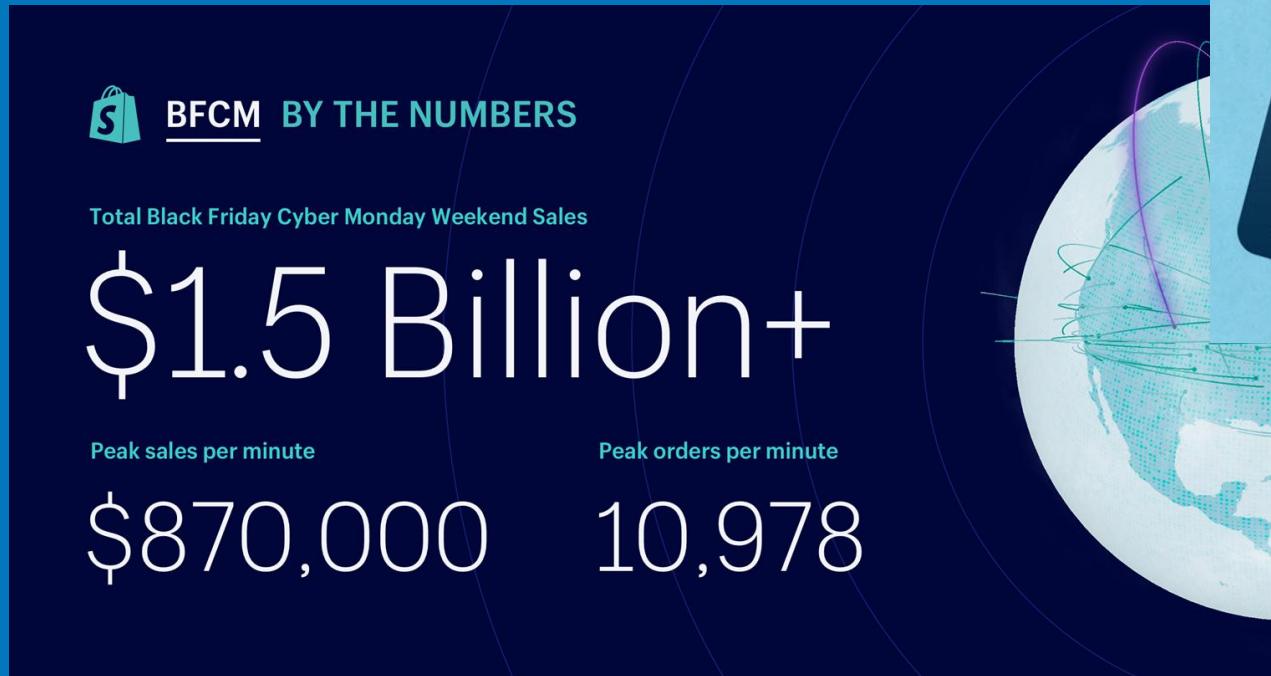


5.1 Introduction:

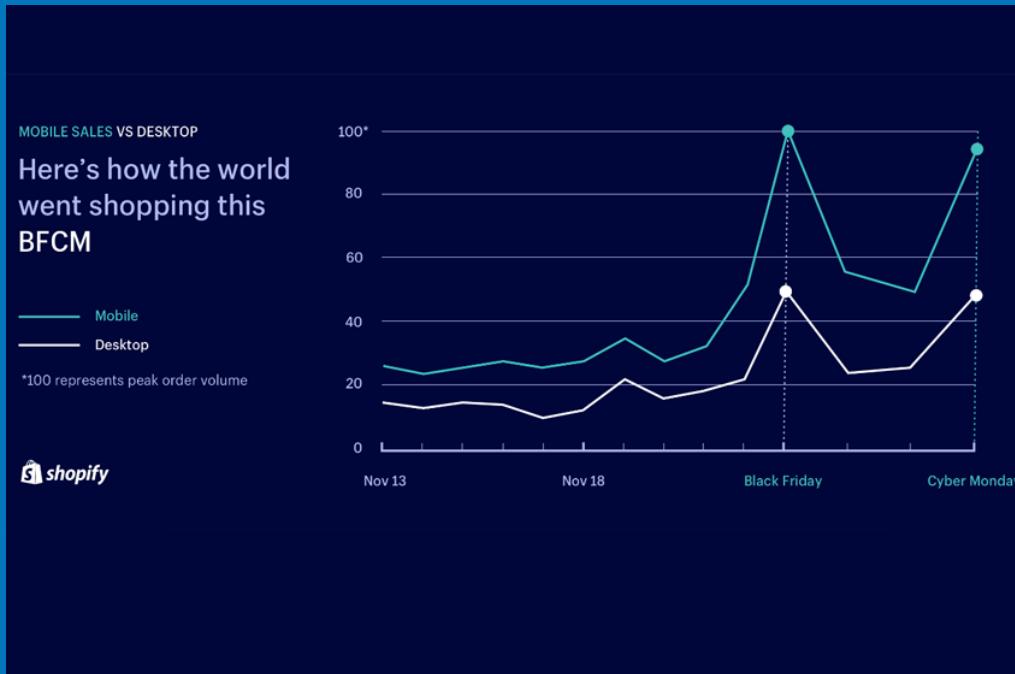
- Requirements
- Planning
- Setup
 - Recording
 - Replaying
 - Testing
 - Staging
 - Production
- Reporting



5.2 Black Friday & Cyber Monday:



5.2 Black Friday & Cyber Monday:



What are the hardware requirements for running a web browser on a PC ?

Component	Usage Range
CPU Cores	1-4+ cores depending on activity.
RAM	200 MB - 8 GB+ depending on tabs and content.
Disk I/O	Minimal, spikes with caching and downloads.
GPU	50 MB - 500 MB+, more with graphics-heavy tasks.
Network Bandwidth	Minimal to 25 Mbps+ for HD/4K streaming.

The hardware footprint of a running web browser on a PC can vary significantly depending on factors like the browser being used, the number and type of open tabs, extensions, and the content being loaded (e.g., static pages vs. resource-intensive applications).

- Client Side Setup:



- OS: Linux/Windows Server 64-bit
- JVM (JMeter now supports JAVA 9) java.com
- Apache JMeter 5.x apache.JMeter.org
- JMeter Plugins (optional)

- The Apache Software Foundation (ASF) is an American non-profit corporation to support Apache software projects, including the Apache HTTP Server. The ASF was formed from the Apache Group in 1999.
- The Apache Software Foundation is a decentralized open source community of developers. The software they produce is distributed under the terms of the Apache License and is **Free and Open-Source Software (FOSS)**.
- Projects: HTTP Server, OpenOffice, Groovy, NetBeans, Maven, JMeter, and many more.



- Apache JMeter is open source software, a 100% pure Java desktop application designed to load test functional behavior and measure performance.

It was originally designed for testing Web Applications but has since expanded to other test functions.

- Protocols:** Web - HTTP, HTTPS; SOAP/REST; FTP; TCP; Database via JDBC; LDAP; Mail - SMTP(S), POP3(S) and IMAP(S); Native commands or shell scripts;

IMAP = Internet Message Access Protocol
JDBC = Java Database Connectivity
LDAP = Lightweight Directory Access Protocol



5.3 Apache JMeter 5:



Run Demo

5.3 Apache JMeter 5:

```
File Edit Selection Find View Goto Tools Project Preferences Help
www.linkedin.com.har ×
1 {
2   "log": {
3     "version": "1.2",
4     "creator": {
5       "name": "WebInspector",
6       "version": "537.36"
7     },
8     "pages": [
9       {
10         "startedDateTime": "2019-06-01T18:04:22.672Z",
11         "id": "page_4",
12         "title": "https://www.linkedin.com/checkpoint/lg/login-submit",
13         "pageTimings": {
14           "onContentLoad": 6603.510999993887,
15           "onLoad": null
16         }
17       }
18     ],
19     "entries": [ ... ]
2080   }
2081 }
2082 }
```



Internet Browser ⇒
Developer Tools ⇒
Network Tab ⇒
Save All as HAR

5.3 Apache JMeter 5:

```
File Edit Selection Find View Goto Tools Project Preferences Help  
www.linkedin.com.har x  
19     "entries": [  
20         {  
21             "startedDateTime": "2019-06-01T18:04:22.671Z",  
22             "time": 1279.0729999542236,  
23             "request": { ...  
282             },  
283             "response": { ...  
539             },  
540             "cache": {},  
541             "timings": { ...  
550             },  
551             "serverIPAddress": "185.63.145.1",  
552             "initiator": {  
553                 "type": "other"  
554             },  
555             "_priority": "VeryHigh",  
556             "_resourceType": "other",  
557             "connection": "114244",  
558             "pageref": "page_4"  
559         },  
      ]
```



HAR ⇒
log, pages, entries ⇒
requests, responses

5.3 Apache JMeter 5:

```
File Edit Selection Find View Goto Tools Project Preferences Help  
www.linkedin.com.har x  
19     "entries": [  
20         {  
21             "startedDateTime": "2019-06-01T18:04:22.671Z",  
22             "time": 1279.0729999542236,  
23             "request": {  
24                 "method": "POST",  
25                 "url": "https://www.linkedin.com/checkpoint/lg/login-submit",  
26                 "httpVersion": "http/2.0",  
27                 "headers": [ ...  
92                     ],  
93                     "queryString": [],  
94                     "cookies": [ ...  
221                     ],  
222                     "headersSize": -1,  
223                     "bodySize": 5698,  
224                     "postData": { ...  
281                         }  
282                     },  
283             }]
```



request ⇒
header,
cookies,
Body (postData),

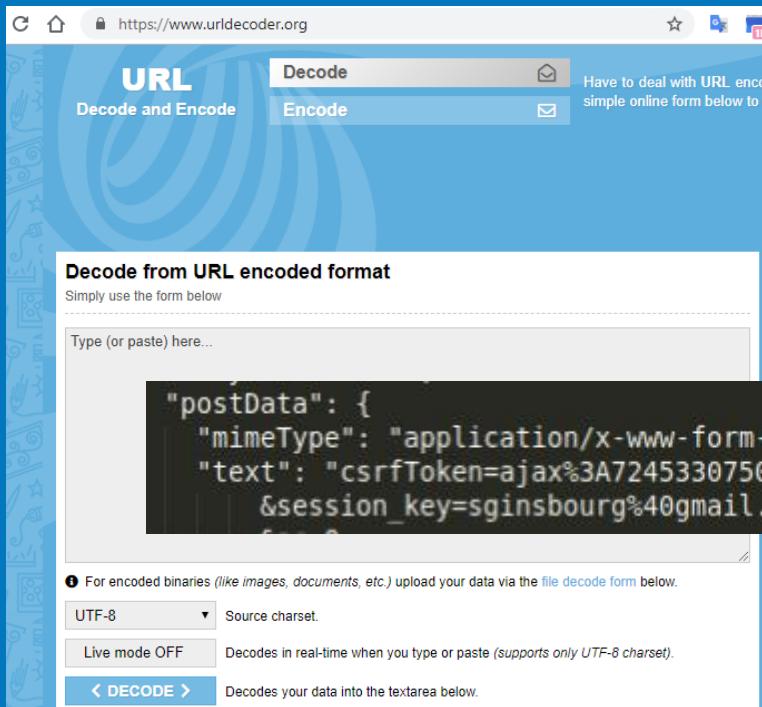
5.3 Apache JMeter 5:

```
File Edit Selection Find View Goto Tools Project Preferences Help
www.linkedin.com.har "method": "POST", ●
1   "method": "POST",
2   "url": "https://www.linkedin.com/checkpoint/lg/login-submit",
3   "httpVersion": "http/2.0",
4   "headers": [
5     "queryString": [],
6     "cookies": [],
7     "headersSize": -1,
8     "bodySize": 5698,
9     "postData": {
10       "mimeType": "application/x-www-form-urlencoded",
11       "text": "csrfToken=ajax%3A7245330750985769141
12         &session_key=sginsbourg%40gmail.com
13         &ac=0
14         &sIdString=22e3813e-4685-4985-855f-d3c750f7d646
15         &controlId=d_checkpoint_lg_consumerLogin-login_submit_button
16         &parentPageKey=d_checkpoint_lg_consumerLogin
17         &pageInstance=urn%3Ali%3Apage%3Ad_checkpoint_lg_consumerLogin%3BztIkhtowR92WKm8kVqZRvw%3D%3D
18         &trk=guest_homepage-basic_nav-header-signin
19         &session_redirect=
20         ...
21       "params": []
```



postData ⇒
token,
session key,
sIdString

5.3 Apache JMeter 5:



csrfToken=ajax%3A724533....

Decode

csrfToken=ajax:724533....

5.3 Apache JMeter 5:

A screenshot of the regex101.com website. The URL in the address bar is `https://regex101.com`. The search term in the input field is `csrfToken=ajax%3A7245330750985769141`. The results page shows a "REGULAR EXPRESSION" section with the pattern `/ csrfToken=ajax%3A\d{10}/`. On the left, there's a sidebar with "FLAVOR" options: PCRE (PHP), ECMAScript, Python, Golang, and "TOOLS" like Code Generator and Regex Debugger.



Pattern ⇒



5.3 Apache JMeter 5:

Recording the business process:



The screenshot shows the Apache JMeter interface. On the left, the Test Plan tree includes nodes for User Defined Variables, HTTP Request Defaults, HTTP Cookie Manager, Thread Group (with Recording Controller), View Results Tree, and two entries under "HTTP(S) Test Script Recorder". The main area displays the "HTTP(S) Test Script Recorder" configuration panel. It has fields for "Name" (set to "HTTP(S) Test Script Recorder"), "Comments", and "State". Below these are "Start", "Stop", and "Restart" buttons. Under "Global Settings", the "Port" is set to 8888 and "HTTPS Domains" is empty. To the right of the recorder panel is a sidebar titled "Configure Proxies to Access the Internet". It contains four radio button options: "No proxy" (unchecked), "Auto-detect proxy settings for this network" (unchecked), "Use system proxy settings" (unchecked), and "Manual proxy configuration:" (checked). Under "Manual proxy configuration:", there are fields for "HTTP Proxy" (localhost) and "Port" (8888), with a checked checkbox "Use this proxy server for all protocols". There are also fields for "SSL Proxy" (localhost) and "Port" (8888).

5.3 Apache JMeter 5:

/ BOI_Load_Test_2019_ver_52_General.jmx (C:\Users\Shay Ginsbourg\Documents\Business\Clients\Tesnet\Bank of Israel\Work\Scripts\V52 - no submit\BOI_Load_Test_

File Edit Search Run Options Tools Help

Test Plan

Name: BOI load test plan 2019

Comments:

Name:	
Website	
protocol	
port	



5.3 Apache JMeter 5:



The screenshot shows the Apache JMeter 5.1.1 interface with the following details:

- Test Plan Tree:** The tree view on the left shows the structure of the test plan:
 - Root node: BOI load test plan 2019
 - Bot (selected)
 - HTTP Cookie Manager
 - HTTP Cache Manager
 - HTTP Request Defaults
 - HTTP Header Manager
 - Portal Ashrai App homepage
 - Portal Ashrai App 108 complaint
 - Portal Ashrai App 108 info
 - Portal Ashrai App 108 tech
 - Portal Ashrai App corporations
 - Aggregate Graph
 - View Results Tree
 - View Results Tree - ERRORS only
 - Debug PostProcessor
- Thread Group Configuration:** The right panel displays the configuration for a Thread Group named "Bot".
 - Name:** Bot
 - Comments:** Action to be taken after a Sampler error
 - Continue
 - Start Next Thread Loop
 - Stop Thread
 - Stop Test
 - Stop Test Now
 - Thread Properties:**
 - Number of Threads (users): 1
 - Ramp-Up Period (in seconds): 1
 - Loop Count: Forever 1
 - Delay Thread creation until needed
 - Scheduler
 - Scheduler Configuration:**

Warning: If Loop Count is not -1 or Forever, duration will be min(Duration, Loop Count * iteration duration)

 - Duration (seconds): []
 - Startup delay (seconds): []

5.3 Apache JMeter 5:

BOI_Load_Test_2019_ver_52_General.jmx (C:\Users\Shay Ginsbourg\Documents\Business\Clients\Tesnet\Bank of Israel\Work\Scripts\V52 - no submit)

File Edit Search Run Options Tools Help

The screenshot shows the Apache JMeter interface. The left pane displays the 'BOI load test plan 2019' tree structure, which includes a 'Bot' node containing various samplers like 'HTTP Cookie Manager', 'HTTP Cache Manager', etc., and several 'Portal Ashrai App' requests. The right pane is focused on the 'HTTP Cookie Manager' configuration dialog. It has fields for 'Name' (set to 'HTTP Cookie Manager'), 'Comments', and an 'Options' section with a checked checkbox for 'Clear cookies each iteration?' and a dropdown for 'Cookie Policy' set to 'standard'. Below this is a 'User-Defined Cookies' table with two columns: 'Name:' and 'Value'.



5.3 Apache JMeter 5:

The image shows the Apache JMeter 5.2 interface. The title bar displays "BOI_Load_Test_2019_ver_52_General.jmx (C:\Users\Shay Ginsbourg\Documents\Business\Clients\Tesnet\Bank of Israel\Work\Scripts\V52 - no submit\BoI)". The menu bar includes File, Edit, Search, Run, Options, Tools, and Help. The toolbar contains various icons for file operations and test elements. The left sidebar shows a tree view of the "BOI load test plan 2019" plan, with a node named "Bot" expanded, listing components like "HTTP Cookie Manager", "HTTP Cache Manager" (which is selected and highlighted in blue), "HTTP Request Defaults", "HTTP Header Manager", and several "Portal Ashrai App" components. A large central window is titled "HTTP Cache Manager". It contains fields for "Name" (set to "HTTP Cache Manager"), "Comments", and two checkboxes: "Clear cache each iteration?" (checked) and "Use Cache-Control/Expires header when processing GET requests" (unchecked). Below these is a "Max Number of elements in cache" field set to "5000". The background of the slide features a large feather logo and the word "ter™".

5.3 Apache JMeter 5:

The screenshot shows the Apache JMeter 5.3 user interface. The left sidebar displays a tree view of the test plan, starting with 'BOI load test plan 2019' which contains a 'Bot' node with various samplers like 'HTTP Cookie Manager', 'HTTP Cache Manager', and 'HTTP Request Defaults'. The 'HTTP Request Defaults' node is currently selected, highlighted in blue. The main panel shows the 'HTTP Request Defaults' configuration dialog. It includes fields for 'Name' (set to 'HTTP Request Defaults'), 'Comments', and tabs for 'Basic' and 'Advanced'. Under the 'Web Server' tab, 'Protocol [http:]' is set to \${protocol}, 'Server Name or IP:' is set to \${Website}, and 'Port Number:' is set to \${port}. The 'HTTP Request' section has a 'Path:' field and tabs for 'Parameters' and 'Body Data'. Below these is a table for 'Send Parameters With the Request' with columns for 'Name:', 'Value', 'URL Encode?', 'Content-Type', and 'Include Equals?'. At the bottom of the dialog are buttons for 'Detail', 'Add', 'Add from Clipboard', 'Delete', 'Up', and 'Down'.



5.3 Apache JMeter 5:

BOI_Load_Test_2019_ver_52_General.jmx (C:\Users\Shay Ginsbourg\Documents\Business\Clients\Tesnet\Bank of Israel\Work\Scripts\V52 - no submit\BOI_Load_Test_2019_ver_52_General.jmx) - Apache JMeter 5.2.1

File Edit Search Run Options Tools Help

BOI load test plan 2019

- Bot
 - HTTP Cookie Manager
 - HTTP Cache Manager
 - HTTP Request Defaults
 - HTTP Header Manager**
 - Portal Ashrai App homepage
 - Portal Ashrai App 108 complaint
 - Portal Ashrai App 108 info
 - Portal Ashrai App 108 tech
 - Portal Ashrai App corporations
- Aggregate Graph
- View Results Tree
- View Results Tree - ERRORS only
- Debug PostProcessor

HTTP Header Manager

Name: HTTP Header Manager

Comments:

Headers Stored in the Header Manager

Name:	Value
User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36

Add Add from Clipboard Delete Load Save



5.3 Apache JMeter 5:

BOI_Load_Test_2019_ver_52_General.jmx (C:\Users\Shay Ginsburg\Documents\Business\Clients\Tesnet\Bank of Israel\Work\Scripts\V52 - no submit\BOI_Load_Test_2019_ver_52_General.jmx) - Apache JMeter 5.2.1

File Edit Search Run Options Tools Help

BOI load test plan 2019

- Bot
 - HTTP Cookie Manager
 - HTTP Cache Manager
 - HTTP Request Defaults
 - HTTP Header Manager
 - Portal Ashrai App homepage**
 - Response Assertion - validation
 - Response Assertion - 200 OK
 - Response Assertion - NOT 404
 - Portal Ashrai App 108 complaint
 - Portal Ashrai App 108 info
 - Portal Ashrai App 108 tech
 - Portal Ashrai App corporations
 - Aggregate Graph
 - View Results Tree
 - View Results Tree - ERRORS only
 - Debug PostProcessor

HTTP Request

Name: Portal Ashrai App homepage

Comments:

Basic | Advanced

Web Server

Protocol [http]: Server Name or IP: Port Number:

HTTP Request

Method: GET Path: /portal-ashrai-app/ Content encoding:

Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data Browser-compatible headers

Parameters | Body Data | Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?

Detail Add Add from Clipboard Delete Up Down



5.3 Apache JMeter 5:



BOI_Load_Test_2019_ver_52_General.jmx (C:\Users\Shay Ginsbourg\Documents\Business\Clients\Tesnet\Bank of Israel\Work\Scripts\V52 - no submit\BOI_Load_Test_2019_ver_52_General.jmx) - Apache JMeter 5.2.1

File Edit Search Run Options Tools Help

BOI load test plan 2019

- Bot
 - HTTP Cookie Manager
 - HTTP Cache Manager
 - HTTP Request Defaults
 - HTTP Header Manager
 - Portal Ashrai App homepage
 - Response Assertion - validation
 - Response Assertion - 200 OK
 - Response Assertion - NOT 404
 - Portal Ashrai App 108 complaint
 - Portal Ashrai App 108 info
 - Portal Ashrai App 108 tech
 - Portal Ashrai App corporations
 - Aggregate Graph
 - View Results Tree
 - View Results Tree - ERRORS only
 - Debug PostProcessor

Response Assertion

Name: Response Assertion - validation
Comments:
Apply to:
 Main sample and sub-samples Main sample only Sub-samples only JMeter Variable Name to use []
Field to Test
 Text Response Response Code Response Message Response Headers
 Request Headers URL Sampled Document (text) Ignore Status
 Request Data
Pattern Matching Rules
 Contains Matches Equals Substring Not Or
Patterns to Test
1 <title>PortalAshraiApp</title>
Add Add from Clipboard Delete
Custom failure message

5.3 Apache JMeter 5:

BOI_Load_Test_2019_ver_52_General.jmx (C:\Users\Shay Ginsbourg\Documents\Business\Clients\Tesnet\Bank of Israel\Work\Scripts\V52 - no submit\BOI_Load_Test_2019_ver_52_General.jmx) - Apache JMeter 5.2.1

File Edit Search Run Options Tools Help

BOI load test plan 2019

- Bot
 - HTTP Cookie Manager
 - HTTP Cache Manager
 - HTTP Request Defaults
 - HTTP Header Manager
 - Portal Ashrai App homepage
 - Response Assertion - validation
 - Response Assertion - 200 OK
 - Response Assertion - NOT 404
 - Portal Ashrai App 108 complaint
 - Portal Ashrai App 108 info
 - Portal Ashrai App 108 tech
 - Portal Ashrai App corporations
 - Aggregate Graph
 - View Results Tree
 - View Results Tree - ERRORS only
 - Debug PostProcessor

Response Assertion

Name: Response Assertion - 200 OK

Comments:

Apply to:
 Main sample and sub-samples Main sample only Sub-samples only JMeter Variable Name to use []

Field to Test

Text Response Response Code Response Message Response Headers
 Request Headers URL Sampled Document (text) Ignore Status
 Request Data

Pattern Matching Rules

Contains Matches Equals Substring Not Or

Patterns to Test

Patterns to Test

1 200

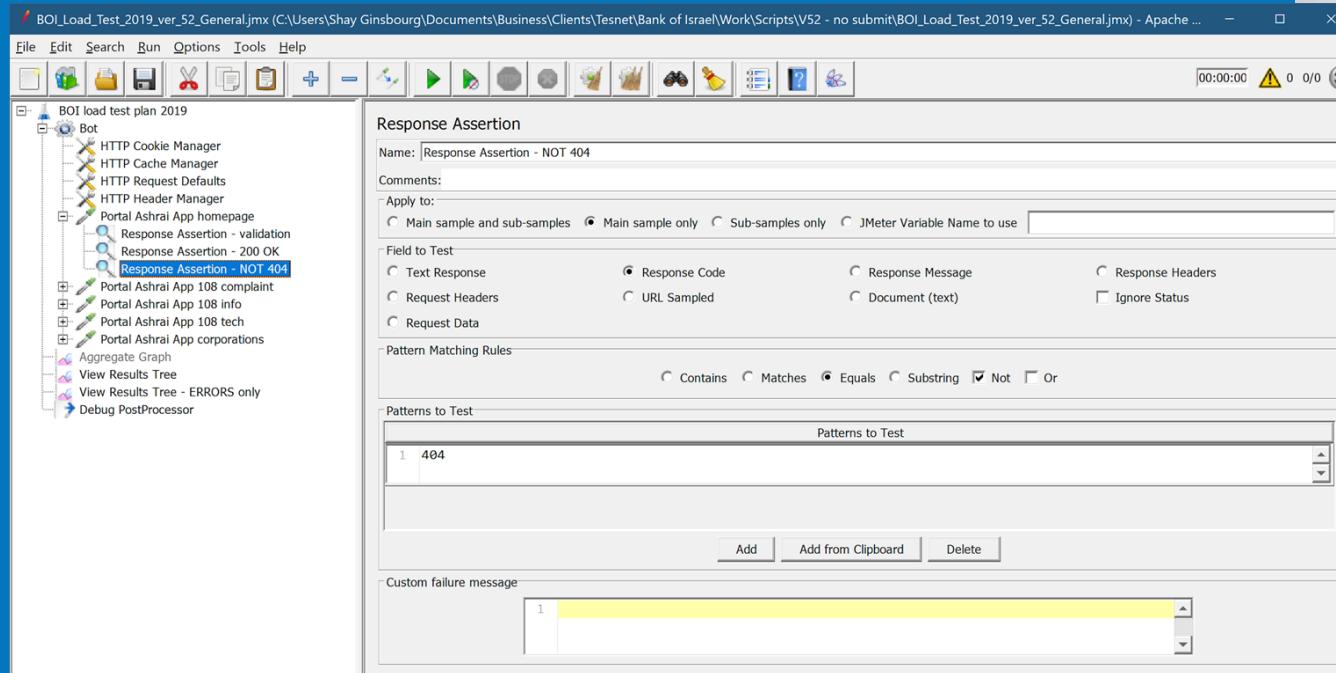
Add Add from Clipboard Delete

Custom failure message

1



5.3 Apache JMeter 5:



The screenshot shows the Apache JMeter 5.2.1 interface with the following details:

- Title Bar:** BOI_Load_Test_2019_ver_52_General.jmx (C:\Users\Shay Ginsbourg\Documents\Business\Clients\Tesnet\Bank of Israel\Work\Scripts\V52 - no submit\BOI_Load_Test_2019_ver_52_General.jmx) - Apache JMeter
- Toolbar:** Includes icons for File, Edit, Search, Run, Options, Tools, Help, and various test elements like Thread Group, Timer, and Assertions.
- Left Panel (Tree View):** BOI load test plan 19
Bot
 - HTTP Cookie Manager
 - HTTP Cache Manager
 - HTTP Request Defaults
 - HTTP Header Manager
 - Portal Ashrai App homepage
 - Response Assertion - validation
 - Response Assertion - 200 OK
 - Response Assertion - NOT 404** (selected)
 - Portal Ashrai App 108 complaint
 - Portal Ashrai App 108 info
 - Portal Ashrai App 108 tech
 - Portal Ashrai App corporations
 - Aggregate Graph
 - View Results Tree
 - View Results Tree - ERRORS only
 - Debug PostProcessor
- Right Panel (Detail View):** Response Assertion settings for "Response Assertion - NOT 404".
 - Name:** Response Assertion - NOT 404
 - Comments:** (empty)
 - Apply to:** Main sample and sub-samples (radio button selected)
 - Field to Test:** Response Code (radio button selected)
 - Pattern Matching Rules:** Equals (radio button selected)
 - Patterns to Test:** 1 404
 - Custom failure message:** (empty)



5.3 Apache JMeter 5:

BOI_Load_Test_2019_ver_52_General.jmx (C:\Users\Shay Ginsburg\Documents\Business\Clients\Tesnet\Bank of Israel\Work\Scripts\V52 - no submit\BOI_Load_Test_2019_ver_52_General.jmx) - Apache JMeter (5.1.1 r1855137)

File Edit Search Run Options Tools Help

BOT load test plan 2019

- HTTP Cookie Manager
- HTTP Cache Manager
- HTTP Request Defaults
- HTTP Header Manager
- Portal Ashrai App homepage
- Response Assertion - validation
- Response Assertion - 200 OK
- Response Assertion - NOT 404
- Portal Ashrai App 108 complaint
- Portal Ashrai App 108 info
- Portal Ashrai App 108 tech
- Portal Ashrai App corporations

Aggregate Graph

Name: Aggregate Graph

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
TOTAL	0	0	0	0	0	0	922337203685...	-92233720368...	0.00%	.0/hour	0.00	0.00

Settings | Graph | Display Graph | Save Graph | Save Table Data | Save Table Header

Column settings

Columns to display: Average Median 90% Line 95% Line 99% Line Min Max Foreground color

Value font: Sans Serif Size: 10 Style: Normal Draw outlines bar Show number grouping? Value labels vertical?

Column label selection: Apply filter Case sensitive Regular exp.

Title

Graph title: Synchronize with name

Font: Sans Serif Size: 16 Style: Bold

Graph size

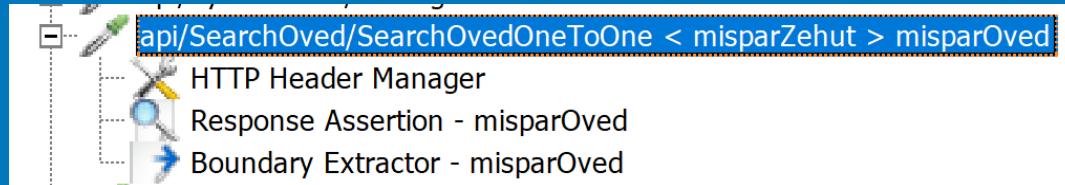
Dynamic graph size Width: Height:

X Axis Max length of x-axis label: Y Axis (milli-seconds) Scale maximum value:

Legend Placement: Bottom Font: Sans Serif Size: 10 Style: Normal



5.3 Apache JMeter 5:



HTTP Request

Name: api/SearchOved/SearchOvedOneToOne < misparZehut > misparOved

Comments:

Basic | Advanced

Web Server

Protocol [http]: \${server5}

HTTP Request

Method: GET Path: /api/SearchOved/SearchOvedOneToOne

Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data Browser-compatible headers

Parameters | Body Data | Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?
misparZehut	\${misparZehut}	<input type="checkbox"/> text/plain

5.3 Apache JMeter 5:



Response Assertion

Name: Response Assertion - misparOved

Comments:

Apply to:

- Main sample and sub-samples
- Main sample only
- Sub-samples only
- JMeter Variable Name to use

Field to Test

- | | | | |
|--|-------------------------------------|--|--|
| <input checked="" type="radio"/> Text Response | <input type="radio"/> Response Code | <input type="radio"/> Response Message | <input type="radio"/> Response Headers |
| <input type="radio"/> Request Headers | <input type="radio"/> URL Sampled | <input type="radio"/> Document (text) | <input type="checkbox"/> Ignore Status |
| <input type="radio"/> Request Data | | | |

Pattern Matching Rules

- Contains
- Matches
- Equals
- Substring
- Not
- Or

Patterns to Test

Patterns to Test

1 misparOved

5.3 Apache JMeter 5:



Regular Expression Extractor

Name:

Comments:

Apply to: Main sample and sub-samples Main sample only Sub-samples only JMeter Variable Name to use

Field to check: Body Body (unesCAPed) Body as a Document Response Headers Request Headers URL Response Code Response Message

Name of created variable:

Regular Expression:

Template (\$i\$ where i is capturing group number, starts at 1):

Match No. (0 for Random):

Default Value: Use empty default value

5.3 Apache JMeter 5:



Boundary Extractor

Name:

Comments:

Apply to: Main sample and sub-samples Main sample only Sub-samples only JMeter Variable Name to use

Field to check: Body Body (unescape) Body as a Document Response Headers Request Headers URL Response Code Response Message

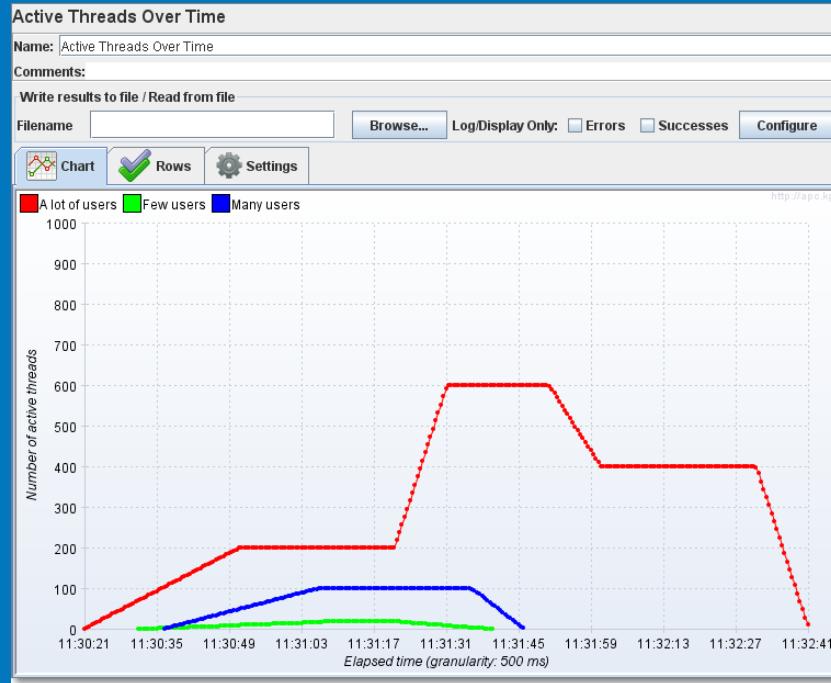
Name of created variable:

Left Boundary: misparOved":

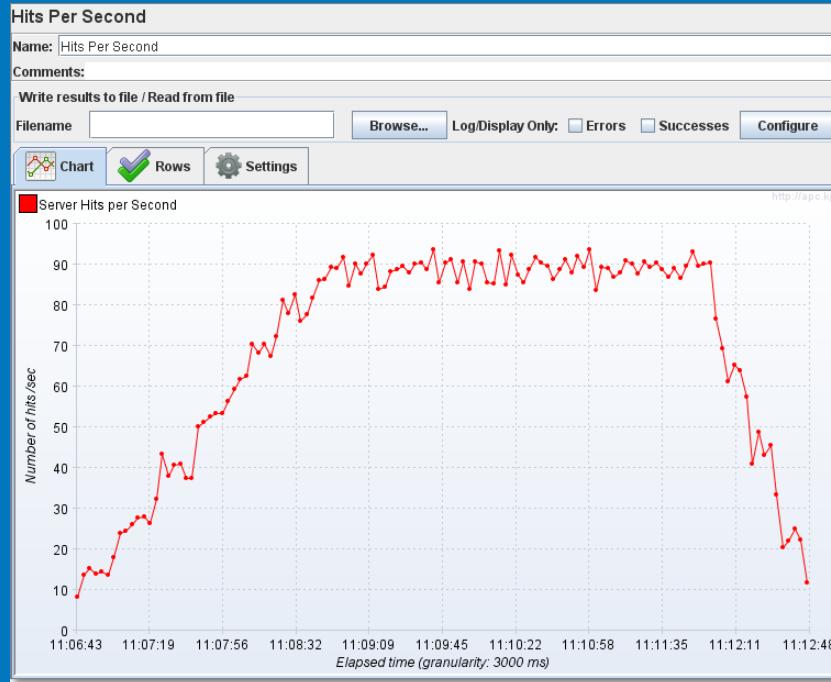
Right Boundary:

Match No. (0 for Random):

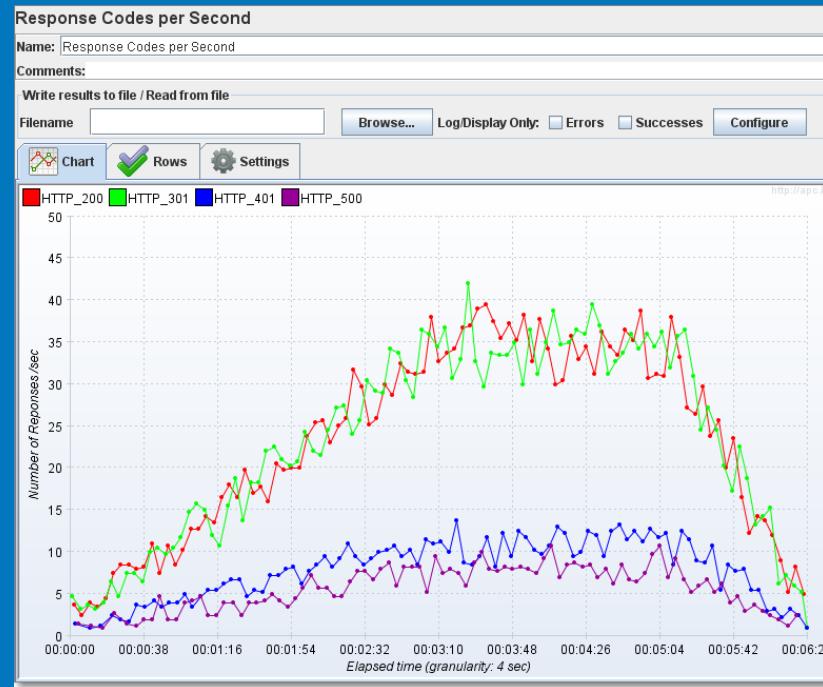
Default Value: Use empty default value



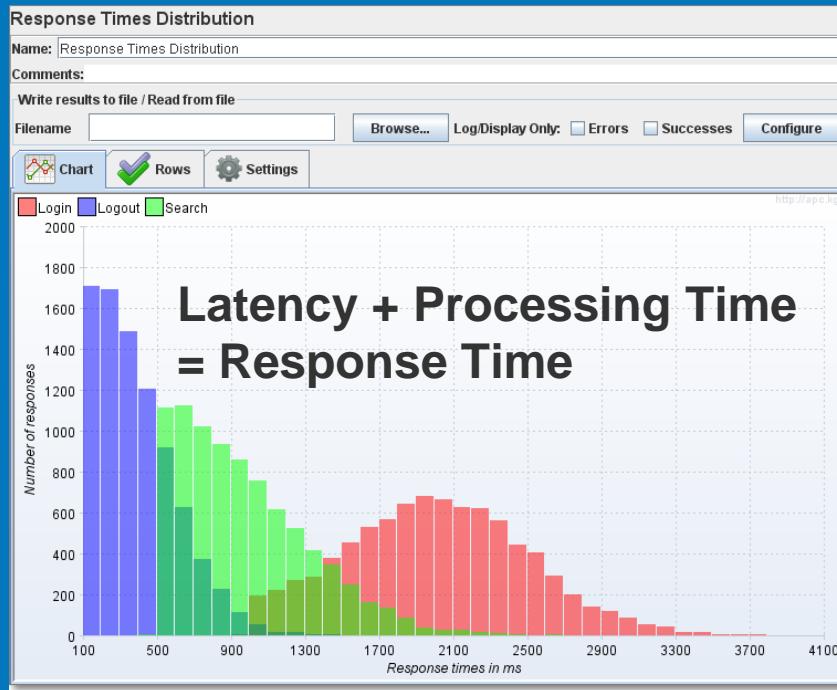
Active Threads Over Time is a simple listener showing how many active threads are there in each thread group during test run.



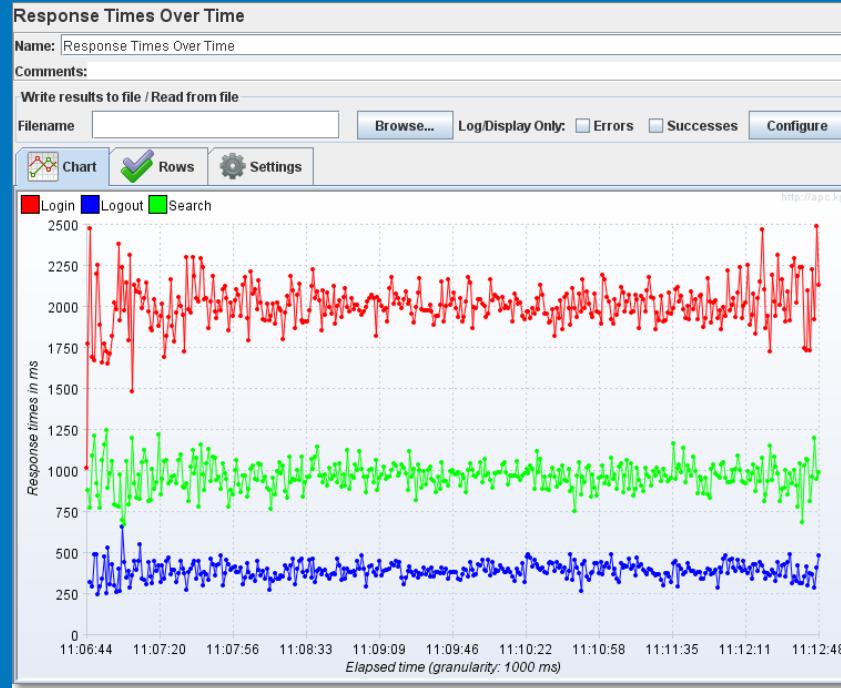
This graph displays the **hits per second** generated by the test plan to the server. Hits include child samples from transactions and embedded resources hits.



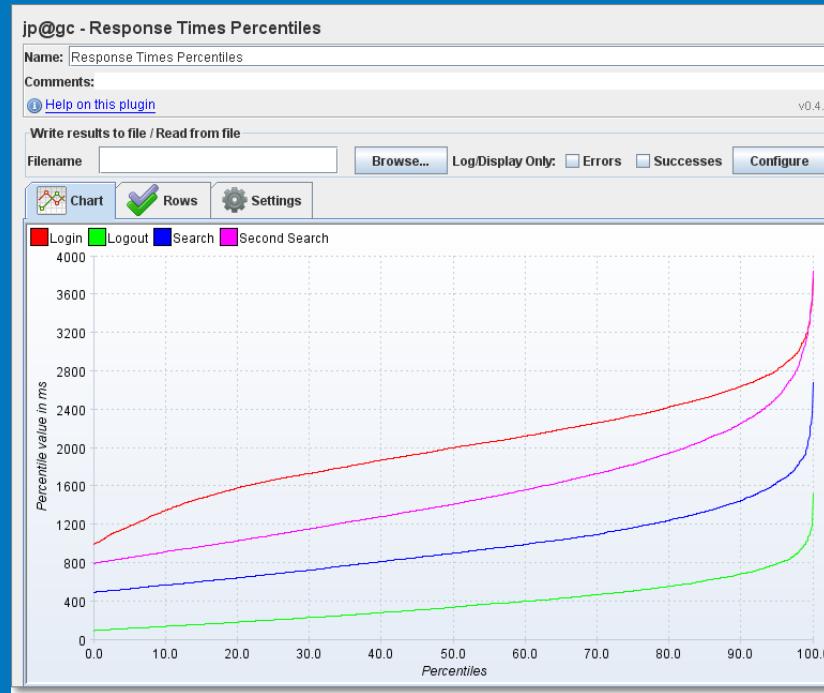
This graph displays the **response code per second** returned from the server during the load test.



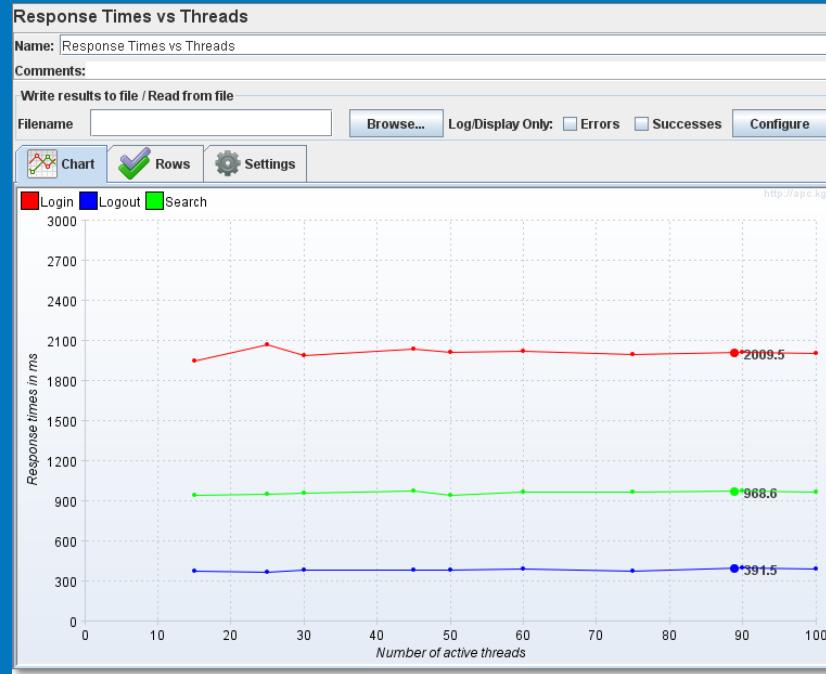
This graph displays the **response time distribution** during the test. The X axis shows the response times grouped by interval, and the Y axis the number of samples which are contained in each interval.



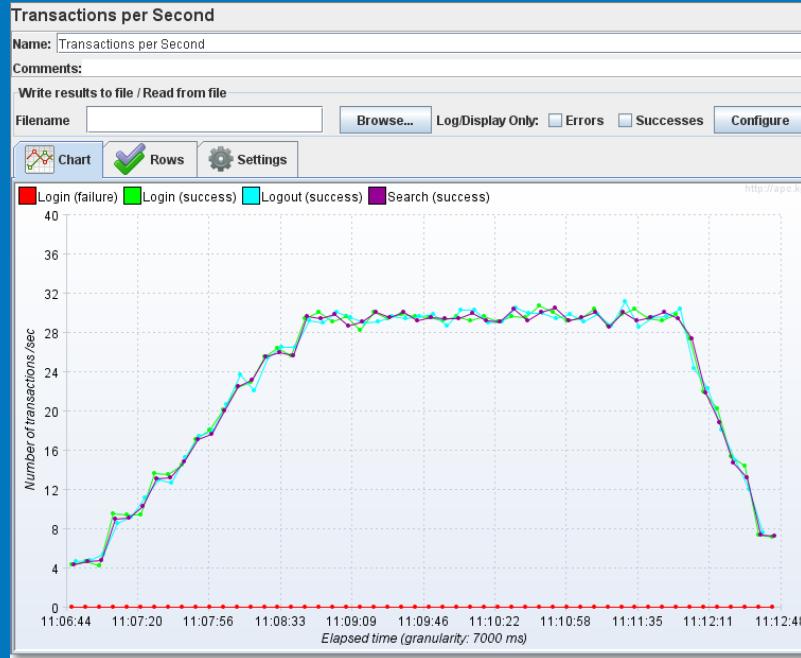
This graph displays for each sampler the **average response time** in milliseconds.



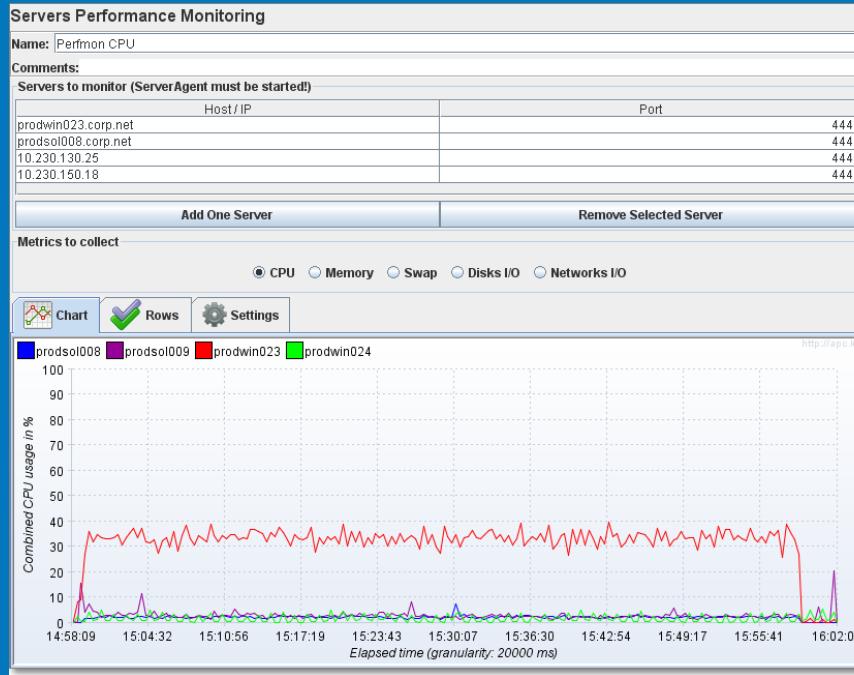
This graph displays the **percentiles for the response time values**. The X Axis represents percentage, and the Y Axis Response time values. One point (P , Value) means for the whole scenario, P percent of the values are below Value ms.



This graph shows how **Response Time** changes with amount of parallel threads. Naturally, server takes longer to respond when a lot of users requests it simultaneously. This graph visualizes such dependencies.



This graph shows the number of **transactions per second** for each sampler. It counts for each seconds the number of finished transactions.



During a load test, it is important to monitor the servers (localhost, Load balancer, web server, app server, database server). The monitored parameters include: **CPU, Memory, Swap, Disks I/O and Networks I/O**.

New and Useful features:



XPath2 Extractor

Name:	XPath2 Extractor
Comments:	
Apply to:	<input type="radio"/> Main sample and sub-samples <input checked="" type="radio"/> Main sample only <input type="radio"/> Sub-samples only <input type="radio"/> JMeter Variable Name to use <input type="text"/>
Name of created variable:	bookTitle
XPath query:	//title/text()
Match No. (0 for Random):	1
Default Value:	bookTitle_NOT_FOUND

New **XPath2 Extractor** allows the user to extract value(s) from structured response - XML or (X)HTML - using XPath2 query language.

New and Useful features:

Sampler result Request Response data

Response Body Response headers

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J. K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
</bookstore>
```



JMeterVariables:
=Shay Ginsbourg
Downloads=C:\Users\Shay Ginsbourg\Dropbox\Studio\l
JMeterThread.last_sample_ok=true
JMeterThread.pack=org.apache.jmeter.threads.SampleF
START.HMS=105852
START.MS=1540108732624
START.YMD=20181021
TESTSTART.MS=1540108747322
_jm_Thread Group_idx_0
bookTitle=Harry Potter
bookTitle_1=Harry Potter
bookTitle_matchNr=1



New **XPath2 Extractor** allows the user to extract value(s) from structured response - XML or (X)HTML - using XPath2 query language.

New and Useful features:



Boundary Extractor

Name: Boundary Extractor

Comments:

Apply to:

Main sample and sub-samples Main sample only Sub-samples only JMeter Variable Name to use

Field to check

Body Body (unesaped) Body as a Document Response Headers Request Headers URL Response Code Response Message

Name of created variable:

Left Boundary:

Right Boundary:

Match No. (0 for Random):

Default Value: Use empty default value

New **Boundary Extractor** element available provides easy extraction with better performances.

New and Useful features:

Sampler result Request Response data

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <title>Welcome to Ginsbourg.com - Ginsbourg.com</title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <meta name="keywords" content="Managed Services">
    <meta name="description" content="Managed Services">
    <!-- master box -->
    <link rel="canonical" href="http://www.ginsbourg.com/">

<meta property="og:site_name" content="Ginsbourg.com"
" />
<meta property="og:title" content="Managed Services" />
```



JMeterVariables:
FakeNews=Managed Services
JMeterThread.last_sample_ok=true

New **Boundary Extractor** element available provides easy extraction with better performances.

New and Useful features:



JSON Extractor

Name:

Comments:

Apply to: Main sample and sub-samples Main sample only Sub-samples only JMeter Variable Name to use

Names of created variables:

JSON Path expressions:

Match No. (0 for Random):

Compute concatenation var (suffix _ALL):

Default Values:

New JSON Assertion element available to assert on JSON responses.

JSON = JavaScript Object Notation

New and Useful features:



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>
  <body>
    <script>

      var jsonData = {
        "name": "Simpsons family",
        "members": [
          {"firstName": "Homer", "lastName": "Simpson"},
          {"firstName": "Marge", "lastName": "Simpson"},
          {"firstName": "Bart", "lastName": "Simpson"},
          {"firstName": "Lisa", "lastName": "Simpson"},
          {"firstName": "Maggie", "lastName": "Simpson"}
        ]
      };

    </script>
  </body>
</html>
```

New JSON Assertion element available to assert on JSON responses.

New and Useful features:



The image shows a split-screen view of JMeter's interface. On the left, the 'Test Plan' tree view is shown with a 'Boundary Extractor > MoreFakeNews' node selected. On the right, a detailed configuration dialog for the 'Boundary Extractor' is displayed, showing fields for Name, Comments, Apply to, Field to check, and various boundary definitions.

Test Plan Tree View (Left):

- Test Plan
- Thread Group
 - HTML 200
 - Boundary Extractor > MoreFakeNews
 - JSON Extractor > JsonFakeNews
 - Debug PostProcessor
 - Precise Throughput Timer
 - Aggregate Graph
 - View Results Tree
 - jp@gc - Response Codes per Second

Boundary Extractor Configuration Dialog (Right):

Name: Boundary Extractor > MoreFakeNews

Comments:

Apply to:

Main sample and sub-samples Main sample only Sub-samples only JMeter

Field to check:

Body Body (unescape) Body as a Document Response Headers Re

Name of created variable: MoreFakeNews

Left Boundary:
var jsonData =

Right Boundary:
;

Match No. (0 for Random): 1

Default Value: MoreFakeNews_NOT_FOUND Use empty default value

New JSON Assertion element available to assert on JSON responses.

New and Useful features:



JSON Extractor

Name: JSON Extractor > JsonFakeNews

Comments:

Apply to:

Main sample and sub-samples Main sample only Sub-samples only JMeter Variable Name to use MoreFakeNews

Names of created variables: JsonFakeNews

JSON Path expressions: \$.name

Match No. (0 for Random): 1

Compute concatenation var (suffix _ALL):

Default Values: JsonFakeNews_NOT_FOUND

New **JSON Assertion** element available to assert on JSON responses.

New and Useful features:



Sampler result	Request	Response data
<pre>JMeterVariables: JMeterThread.last_sample_ok=true JMeterThread.pack=org.apache.jmeter.threads.SamplePackage@43ec346e JsonFakeNews=Simpsons family JsonFakeNews_matchNr=1 MoreFakeNews={ "name": "Simpsons family", "members": [{"firstName": "Homer", "lastName": "Simpson"}, {"firstName": "Marge", "lastName": "Simpson"}, {"firstName": "Bart", "lastName": "Simpson"}, {"firstName": "Lisa", "lastName": "Simpson"}, {"firstName": "Maggie", "lastName": "Simpson"}] }</pre>		

New JSON Assertion element available to assert on JSON responses.

New and Useful features:



[timeShift](#) - return a date in various formats with the specified amount of seconds/minutes/hours/days added.

Function Helper

Choose a function timeShift Help

Function Parameters

Name:	Value
Format string for DateTimeFormatter (optional) (default unix timestamp in millisecond)	dd/MM/yyyy
Date to shift (optional) (default : now)	21/08/2017
Amount of seconds/minutes/hours/days to add (example P2D : plus one day) (optional)	P2D
String format of a locale (ex: fr_FR , en_EN) (optional)	
Name of variable in which to store the result (optional)	

Detail Add Add from Clipboard Delete

Copy and paste function string \${timeShift(dd/MM/yyyy,21/08/2017,P2D,,)}

Generate

The result of the function is

```
1 23/08/2017
```

New and Useful features:



[RandomDate](#) - generate random date within a specific date range.

Choose a function Help

Function Parameters

Name:	Value
Format string for DateTimeFormatter (optional) (default yyyy-MM-dd)	
Start date (optional) (default: now)	2016-09-21
End date	2017-09-21
String format of a locale (ex: fr_FR , en_EN) (optional)	
Name of variable in which to store the result (optional)	

Detail Add Add from Clipboard Delete

Copy and paste function string Generate

The result of the function is

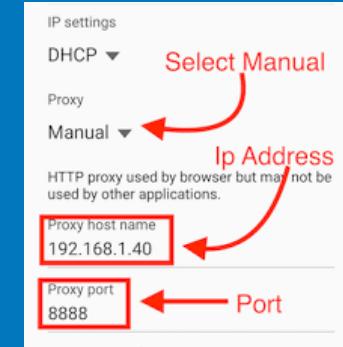
1	2017-01-19
---	------------

Recording scripts from browser on
all types and operating systems:

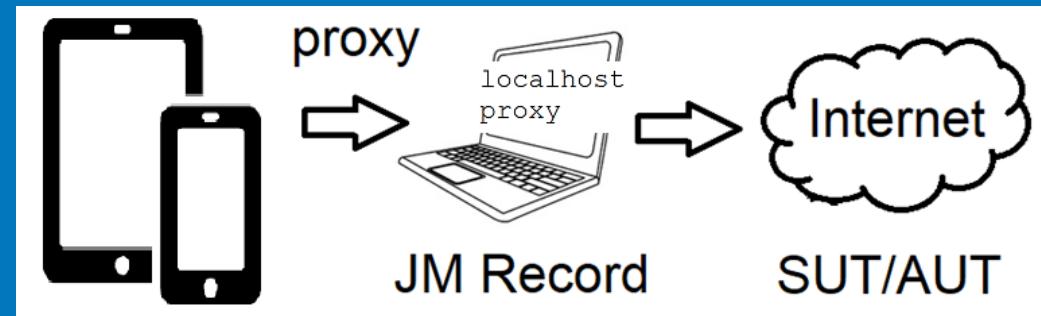
https://JMeter.apache.org/usermanual/JMeter_proxy_step_by_step.html



Recording scripts from mobile applications and tablets of all types and operating systems:



Double proxy approach



Unit 5: Final Project: Step 1

1.1 Install Java 64-bit on Linux or on Windows java.com

1.2 Download Apache JMeter (ZIP) latest version apache.JMeter.org

1.3 Launch Apache JMeter

Unit 5: Final Project: Step 2

2.1 Record a business process with Apache JMeter from one of the websites specified at:

<https://abstracta.us/blog/software-testing/best-demo-websites-for-practicing-different-types-of-software-tests/>

2.2 Convert HAR (HTTP Archive) to JMX (JMeter XML) (optional)

Unit 5: Final Project: Step 3

3.1 Replay the business process with Apache JMeter

After adding to the bot elements, as follows:

Extractors, response code validations and response assertions

3.2 Create an Automatic Correlation for Apache JMeter

Including REGEX (optional)

Unit 5: Final Project: Step 4

4.1 Load a business process with Apache JMeter
10 concurrent bots (virtual users) for 20 minutes

4.2 Generate HTML report with Apache JMeter

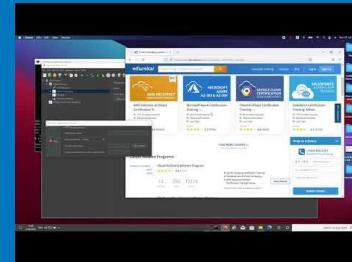
4.3 Export HTML report to PDF

Unit 5: Final Project:

For more guidance:

<https://youtu.be/PXYgHhvU6vY>

By Adham Eldda



Unit 5: EOF



shaygi1@ac.sce.ac.il



