



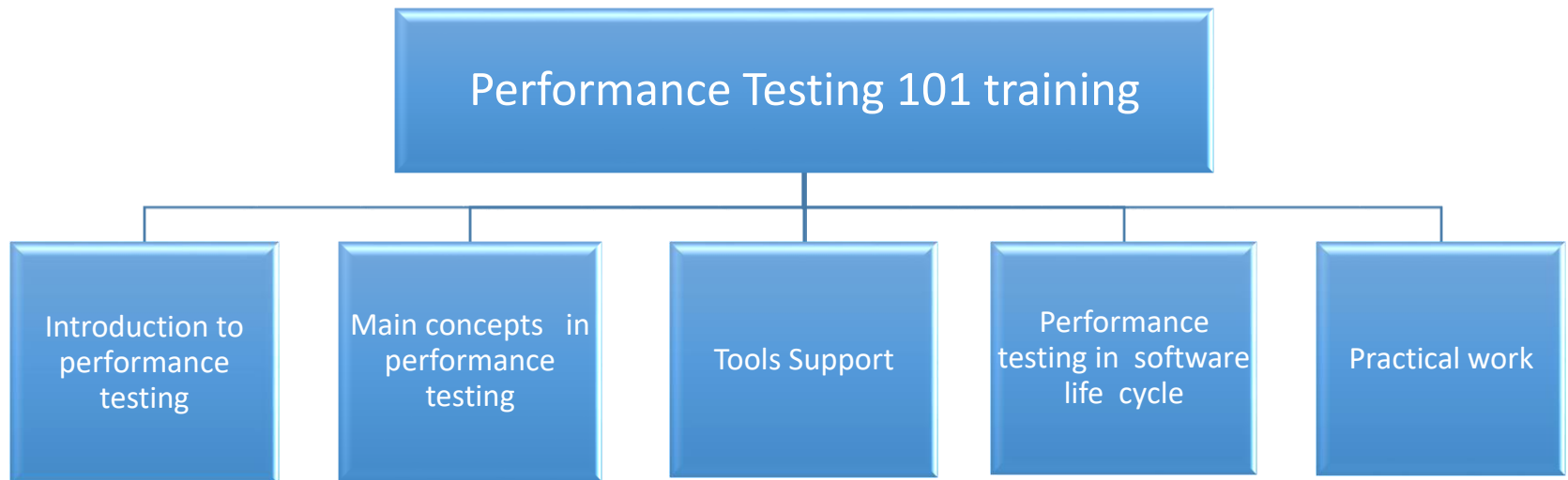
الْجَمْعِيَّةُ الْعِلْمِيَّةُ الْمَلَكِيَّةُ
Royal Scientific Society

Performance Testing 101 Training

RSS-NSQAC

2022

Training overview



Introduction to performance testing

Outlines:

- Types of Software testing.
- What is performance testing.
- Why we need Performance Testing.
- Performance Testing Types.
- Performance metrics.



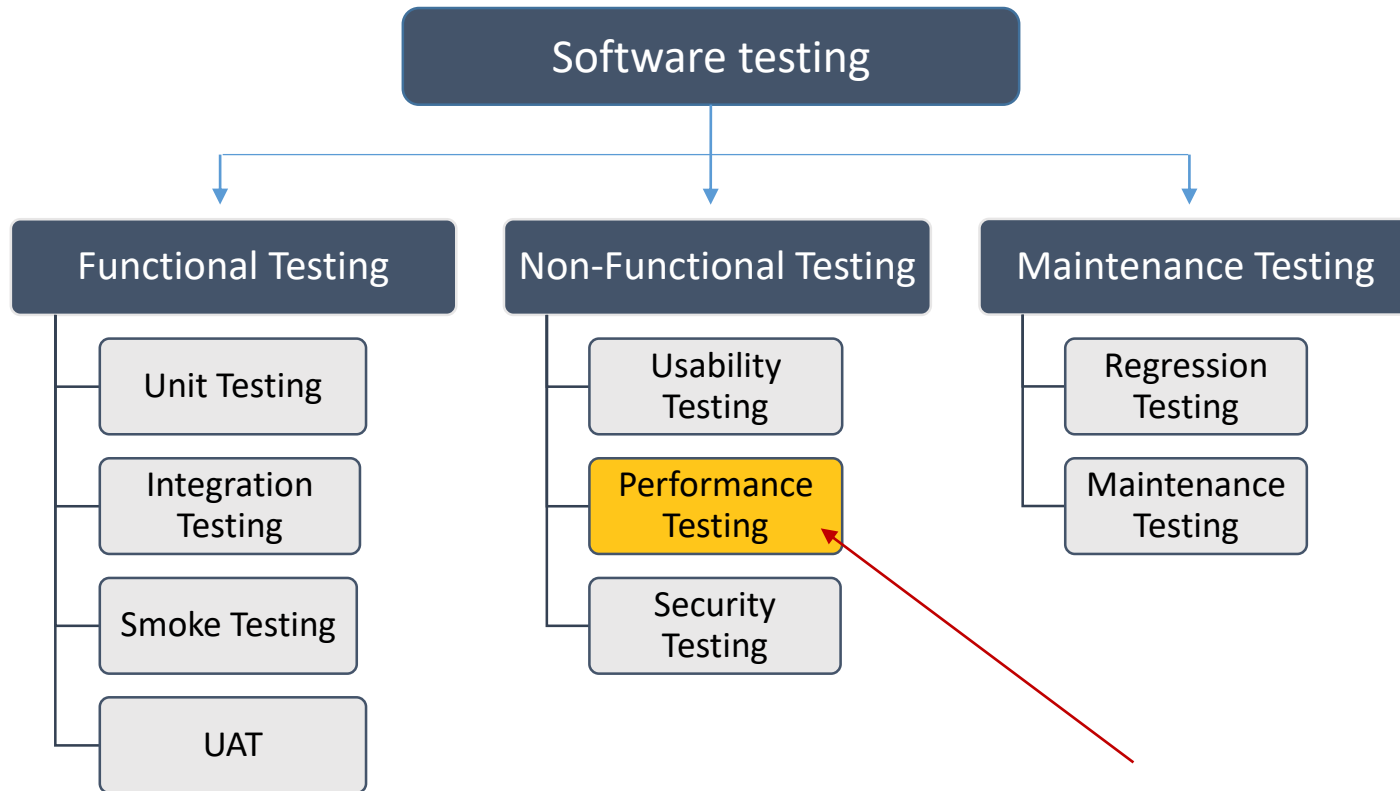
Introduction to performance testing

Outlines:

- Types of Software testing.
- What is performance testing.
- Why we need Performance Testing.
- Performance Testing Types.
- Performance metrics.



Types of Software Testing



Introduction to performance testing

Outlines:

- Types of Software testing.
- What is performance testing.
- Why we need Performance Testing.
- Performance Testing Types.
- Performance metrics.



What is performance testing

- Performance testing is a non-functional software testing technique that determines how the stability, speed, scalability, and responsiveness of an application holds up under a given workload.
- Performance is an essential part of providing a “good experience” for users when they use their applications on a variety of fixed and mobile platforms.
- Performance testing plays a critical role in establishing acceptable quality levels for the end user and is often closely integrated with other disciplines such as usability engineering and performance engineering.



Introduction to performance testing

Outlines:

- Types of Software testing.
- What is performance testing.
- Why we need Performance Testing.
- Performance Testing Types.
- Performance metrics.



Why we need Performance Testing

- Performance testing helps to check the behavior of an application across various situations (a particular **workload** within **specific period of time**).
- A system can work effectively with a specific number of concurrent users, but malfunction might occurs with additional thousands of concurrent users during peak traffic.
- The **goal** of Performance Testing is not to find bugs but to eliminate performance bottlenecks.



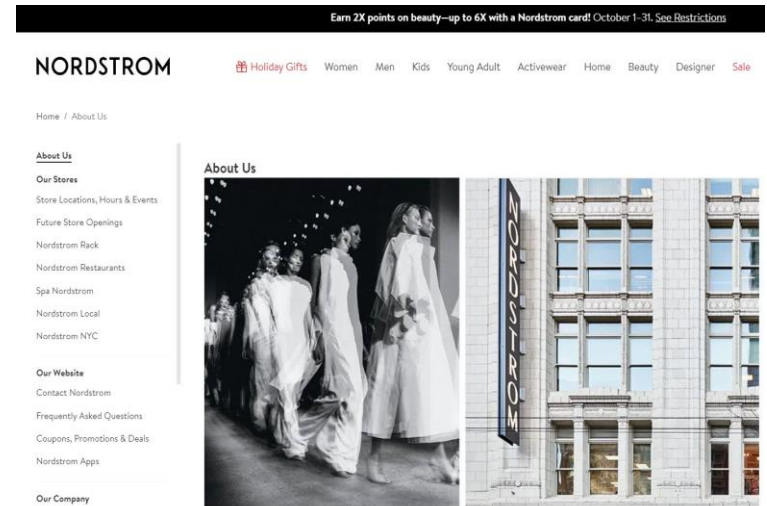
What could happen without good Performance Test

Without Performance Testing, software is likely to suffer from issues such as: running slow while several users use it simultaneously, inconsistencies across different operating systems and poor usability.

- EduWave: “Tawjeehi” results day



- The Nordstrom website crashed on a big shopping day



Introduction to performance testing

Outlines:

- Types of Software testing.
- What is performance testing.
- Why we need Performance Testing.
- Performance Testing Types.
- Performance metrics.



Performance Testing Types

Load
Testing

Stress
Testing

Spike
Testing

Endurance
Testing

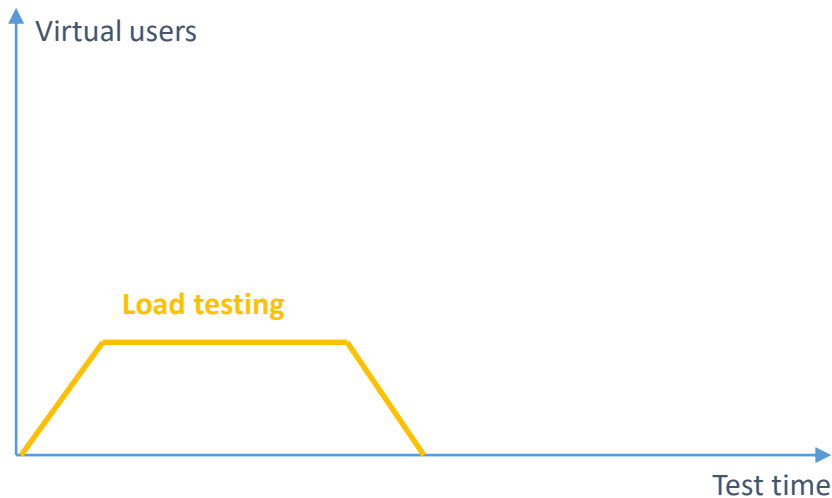
Scalability
Testing

Volume
Testing



Performance Testing Types: Load Testing

- **Load testing** is a type of testing which involves evaluating the performance of the system under the expected workload.
- **Goal:** To verify that the platform can handle the expected number of transactions and to verify its behaviour when the maximum number of concurrent users is reached within a certain time.

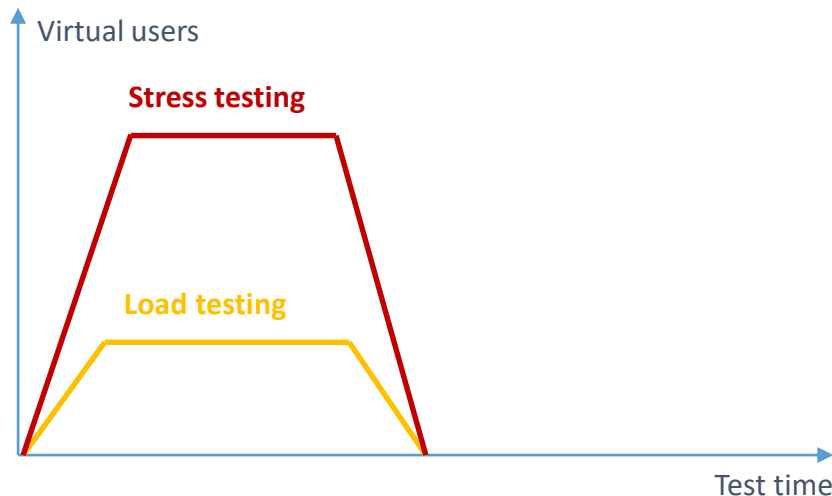


Will my Software be able to handle the expected load levels ?



Performance Testing Types: Stress testing

- **Stress testing** is a type of performance testing where we evaluate the application's performance at load much higher than the expected load.
- **Goal:** To verify the behaviour of the system once the load is increased to more than the system design expectations. This test mainly measures the system on its robustness and error handling capabilities under extremely heavy load conditions

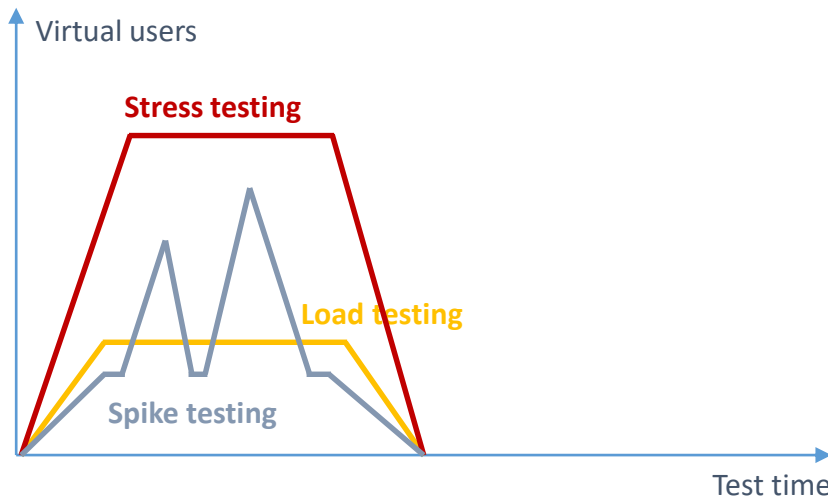


How my system will behave if the load become beyond the expected levels ? How will it handle errors ?



Performance Testing Types: Spike testing

- In **spike testing**, we analyze the behavior of the system on suddenly increasing the numbers of users.
- **Goal:** To check system with extreme increments and decrements in traffic load to evaluate the behaviour of the software application under sudden increment or decrement in user load and determine recovery time after a spike of user load.

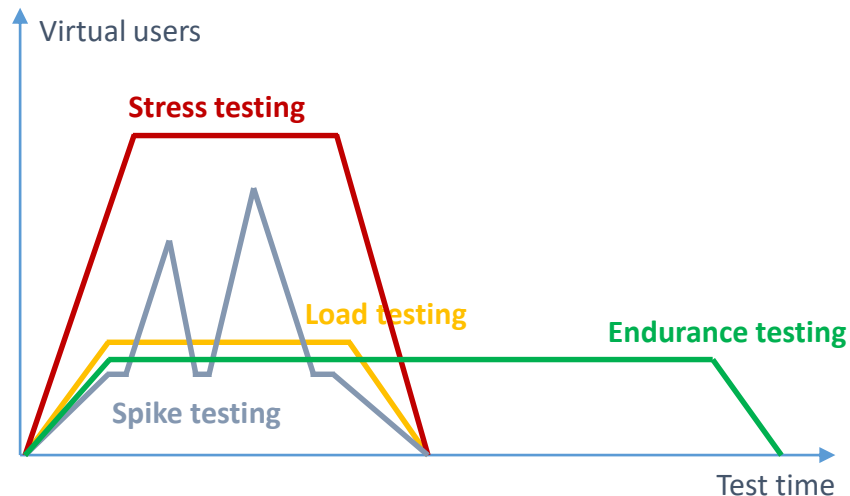


Will my software be able to handle spike loads ? How much recovery time it will need?



Performance Testing Types: Endurance testing

- **Endurance testing** is also known as “Soak testing”. It is done to determine if the system can sustain the continuous expected load for long duration.
- **Goal:** To verify that system can sustain the expected amount of load over a long period of time by verifying that there are no resource capacity problems (e.g., memory leaks, database connections, thread pools) that may eventually degrade performance and cause failures at breaking points.



Is my application capable enough to handle extended load without any deterioration of response time ?



Performance Testing Types: Volume testing

- The **volume testing** is performed by feeding the application with a high volume of data. It is also referred to as “flood testing”.
- **Goal:** To check system performance with increased volume of data in the database that can be done by expand database to bigger size by adding more data in the database to increase the capacity and then perform the test.



How my software will be affected by increasing the volume of data in the database?



Introduction to performance testing

Outlines:

- Types of Software testing.
- What is performance testing.
- Why we need Performance Testing.
- Performance Testing Types.
- Performance metrics.



Performance Metrics

What Are Performance Metrics?

performance testing data allows you to understand the effectiveness of performance testing.

There are two types of data that belong here:

- Measurements are data recorded during testing – for example, how many seconds it takes to respond to the request.
- Metrics are calculations made with the help of specific formulas applied to measurements, such as different kinds of percentages, average indicators, etc.

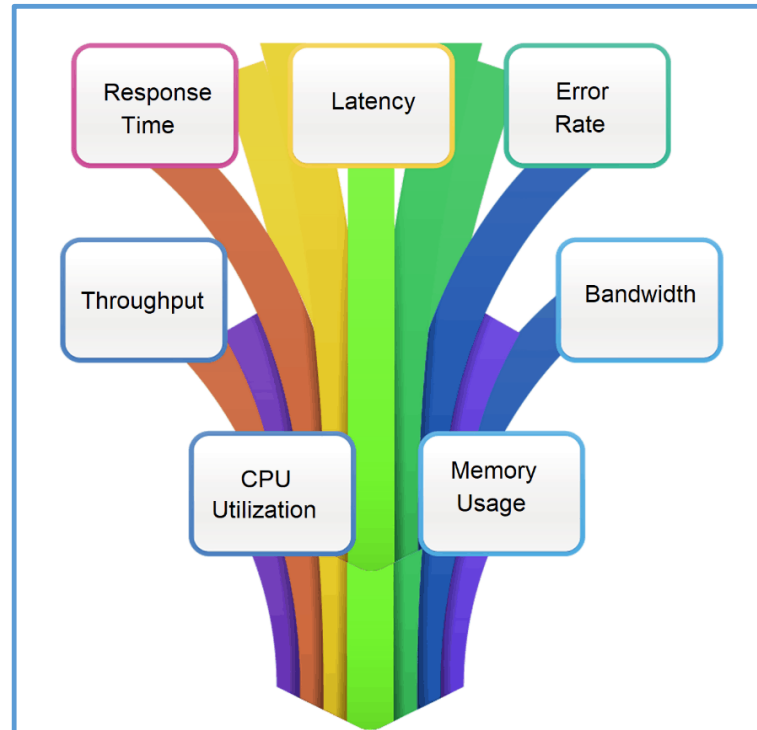
Why Are Performance Metrics Important?

- Metrics are a baseline for the tests.
- They help to track the progress of a project.
- Using metrics, a QA team becomes able to define an issue and measure it for finding a solution.
- Tracking metrics over time allows you to compare the results of tests and estimate the impact of code changes.
- **In general**, it is essential to track performance metrics to define what areas and features require increased attention and quality enhancement.



Performance metrics - Typical Metrics Collected in Performance Testing

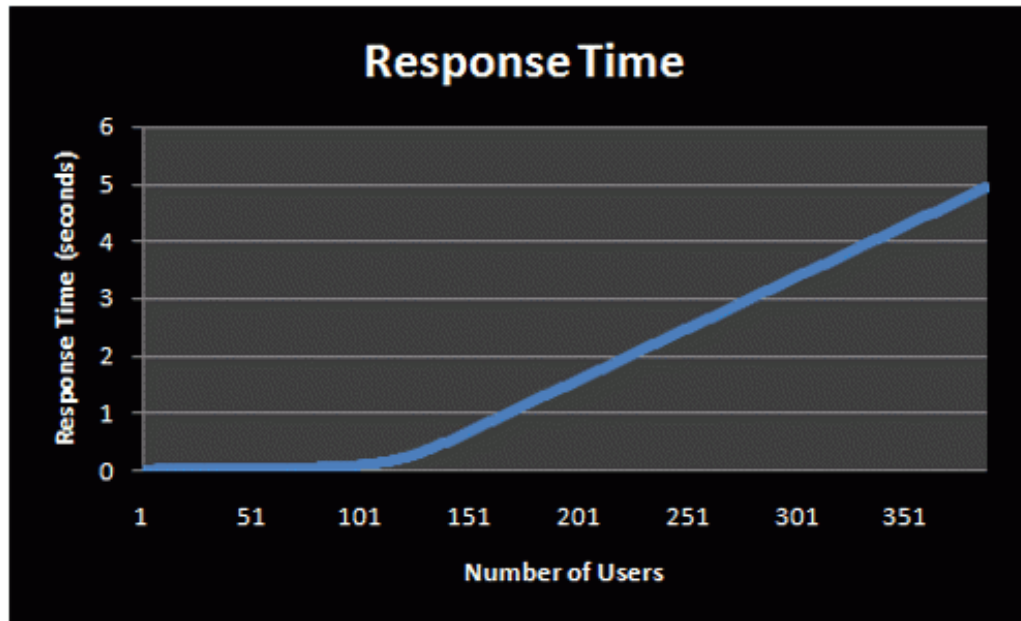
The basic parameters monitored during performance testing include:



Performance metrics - Typical Metrics Collected in Performance Testing

➤ Response Time:

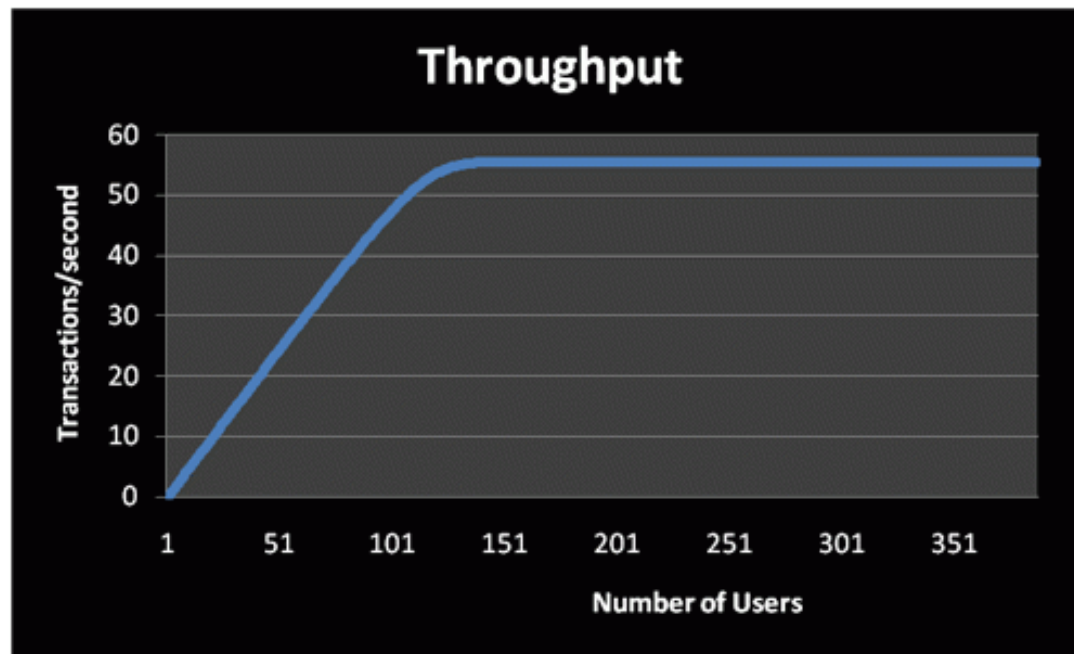
The elapsed time may between a request on a system and the response to that request. And it be determined per transaction, per concurrent user, page load times, etc.



Performance metrics - Typical Metrics Collected in Performance Testing

➤ Throughput:

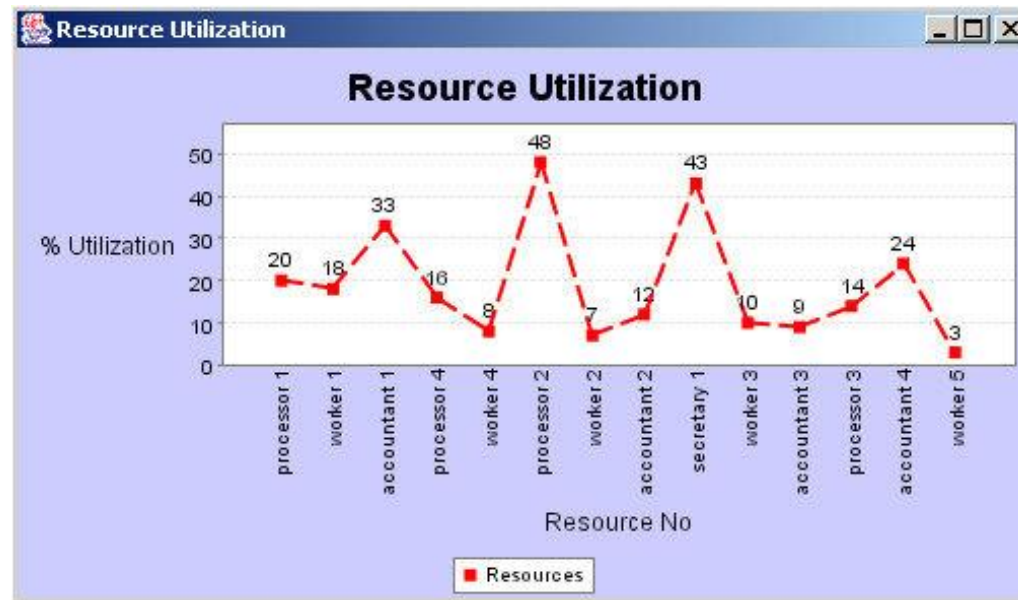
The amount of transactions produced over time during a test. It's also expressed as the amount of capacity that a website or application can handle.



Performance metrics - Typical Metrics Collected in Performance Testing

➤ Resource Utilization:

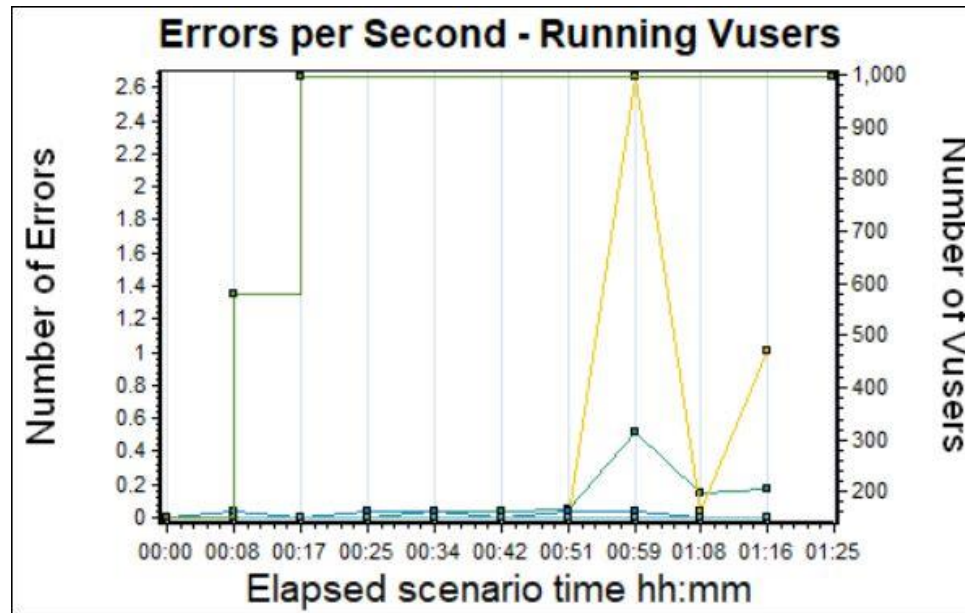
Is a way to track how busy various resources of a computer system are when running a performance test. And the common utilization performance metrics to monitor are: CPU, Memory, Disk and Network.



Performance metrics - Typical Metrics Collected in Performance Testing

➤ Error Rate:

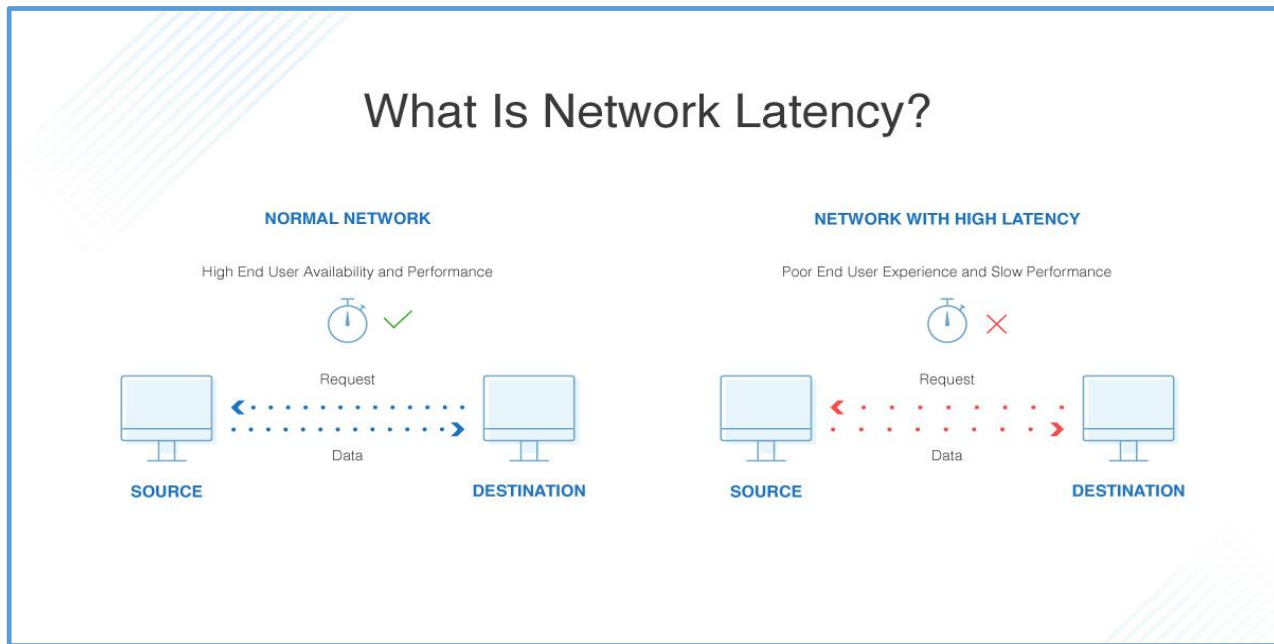
Is the number of errors compared to all requests made during a given performance test cycle. Error Rate is a significant metric because it measures “performance failure” in the application



Performance metrics - Typical Metrics Collected in Performance Testing

➤ Latency:

Time taken for information to get to its target and back again. Sometimes, latency means “delay” which is a very problematic issue when working with remote data centers.



Performance Metrics

How to Select Performance Metrics Correctly?

At first we should establish the goals, then ask questions to know when the goals have been achieved. Metrics are associated with each question to ensure the answer to the question is measurable.

- Specify a client's business objectives to come up with an ultimate list of performance requirements.
- Every feature should have a specific success metric assigned to it – a single parameter or a narrow range of parameters.
- Metrics should correlate with the value delivered to a software user – high speed, software stability, all the features working, etc.



Performance Metrics

There are three key sources of performance metrics:

➤ **Performance Test Tools**

Tools may vary in the number of metrics shown, the way in which the metrics are shown, and the ability for the user to customize the metrics to a particular situation.

Some tools collect and display performance metrics in text format, while more robust tools collect and display performance metrics graphically in a dashboard format.

➤ **Performance Monitoring Tools**

monitoring tools may be used to monitor system performance on an ongoing basis and to alert system administrators to lowered levels of performance and higher levels of system errors and alerts.

➤ **Log Analysis Tools**

These tools scan server logs and compile metrics from them. Some of these tools can create charts to provide a graphical view of the data.



Main concepts in performance testing

Outlines:

- Principles of Performance Testing.
- Load generation concept.
- Aggregating performance testing results.
- Common Performance Efficiency Failure Modes and their causes.
- Performance risks for different architecture.



Main concepts in performance testing

Outlines:

- Principles of Performance Testing.
- Load generation concept.
- Aggregating performance testing results.
- Common Performance Efficiency Failure Modes and their causes.
- Performance risks for different architecture.



Principles of Performance Testing

- Tests must be aligned to the defined expectations of different stakeholder users, system designers and operations staff.
- The groups, in particular tests must be reproducible. Statistically identical results (within a specified tolerance) must be obtained by repeating the tests on an unchanged system.
- The tests must yield results that are both understandable and can be readily compared to stakeholder expectations.
- The tests can be conducted, where resources allow, either on complete or partial systems or test environments that are representative of the production system.
- The tests must be practically affordable and executable within the timeframe set by the project.



Main concepts in performance testing

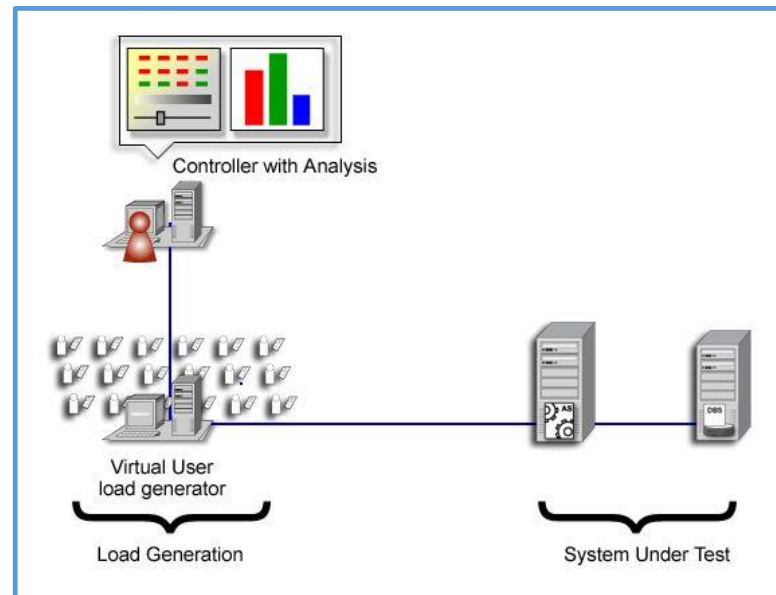
Outlines:

- Principles of Performance Testing.
- Load generation concept.
- Aggregating performance testing results.
- Common Performance Efficiency Failure Modes and their causes.
- Performance risks for different architecture.



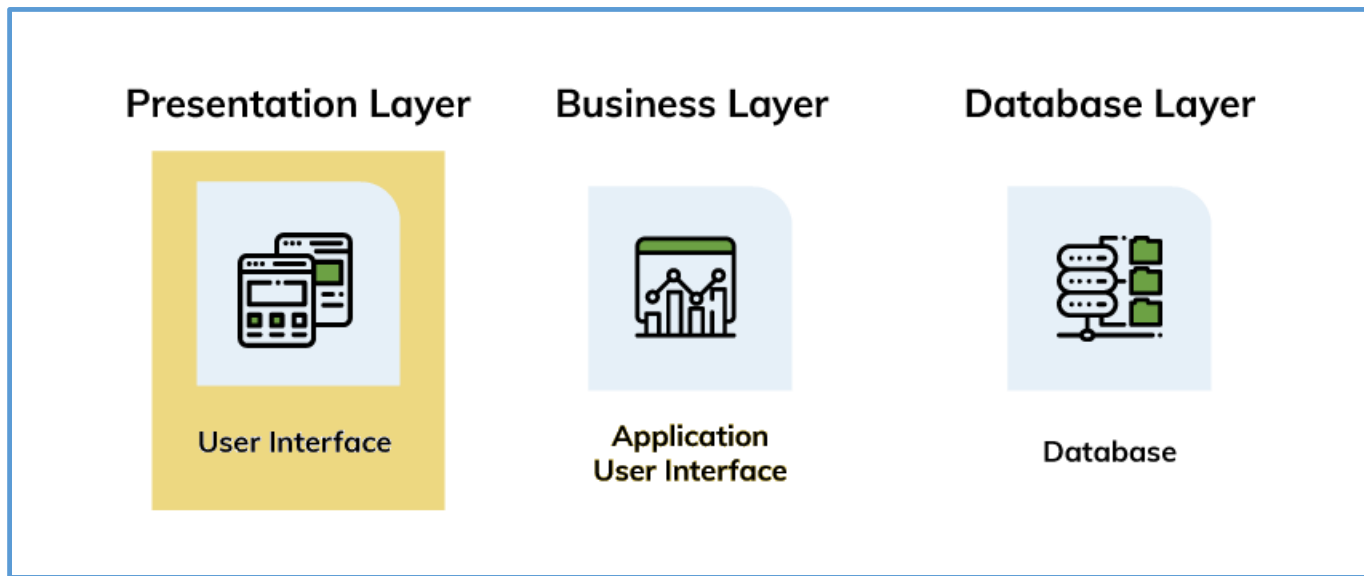
Load Generation Definition

- **Load Generator** is the process of simulating a defined set of activities at a specified load to be submitted to a component or system.
- Load Generator is **used** to generate load on the server to test it for performance and scalability.



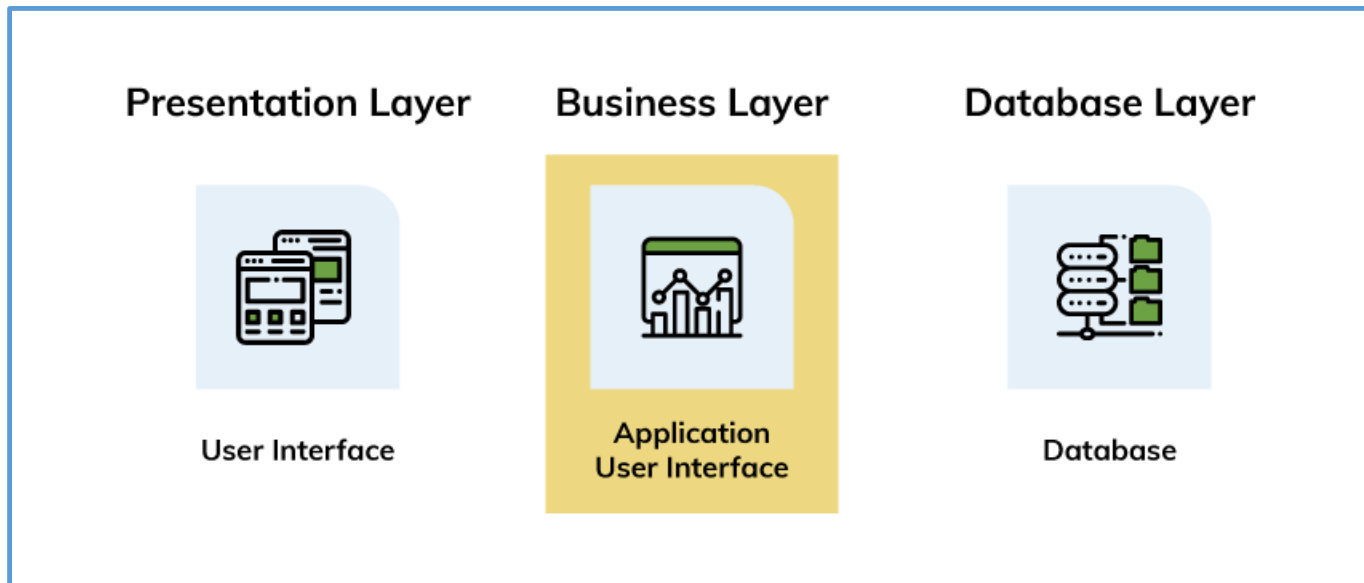
Load Generation Options

- **Load Generation via the User Interface:** This approach may be used if only a small number of users are to be represented and if the required numbers of software clients are available from which to enter required inputs.



Load Generation Options

- **Load Generation via the Application Programming Interface (API):** This approach is similar to using the UI for data entry, but uses the application's API instead of the UI to simulate user interaction with the system under test.



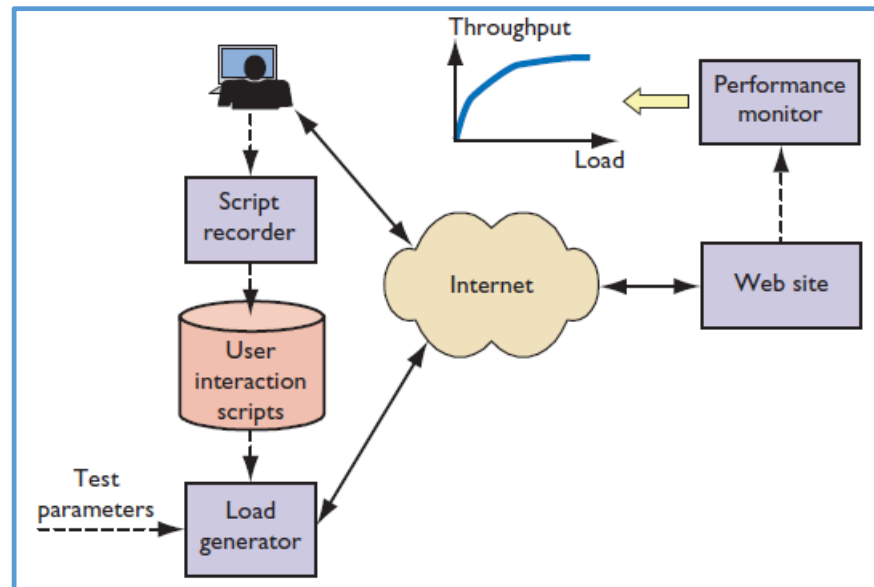
Load Generation Options

- **Load Generation using Crowds:** This approach depends on the availability of a large number of testers who will represent real users. This may be a suitable method for testing applications that are reachable from anywhere in the world.



Load Generation Options

- **Load Generation using Captured Communication Protocols:** This approach involves capturing user interaction with the system under test at the communications protocol level and then replaying these scripts to simulate potentially very large numbers of users in a repeatable and reliable manner.



Main concepts in performance testing

Outlines:

- Principles of Performance Testing.
- Load generation concept.
- Aggregating performance testing results.
- Common Performance Efficiency Failure Modes and their causes.
- Performance risks for different architecture.



Aggregating performance testing results

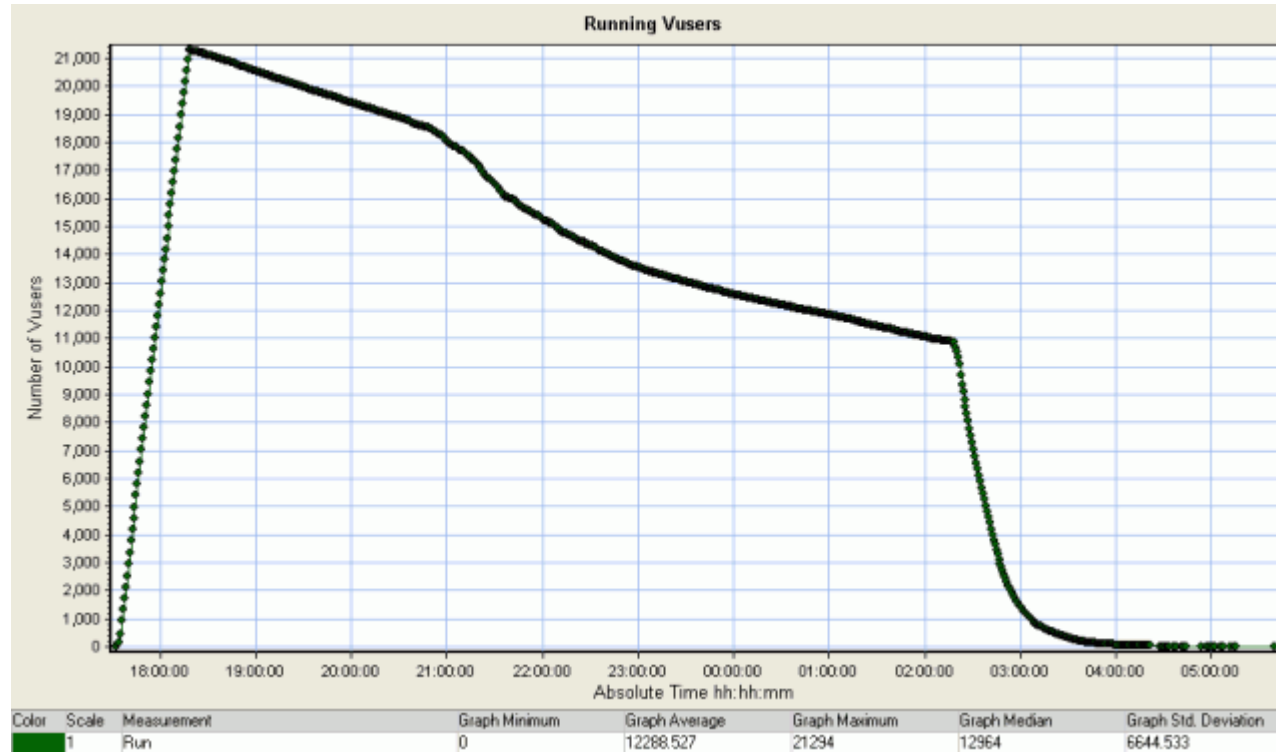
The purpose of aggregating performance metrics is to be able to understand and express them in a way that accurately conveys the total/big picture of system performance.

When performance metrics are viewed at only the detailed level, drawing the right conclusion may be difficult - especially for business stakeholders.



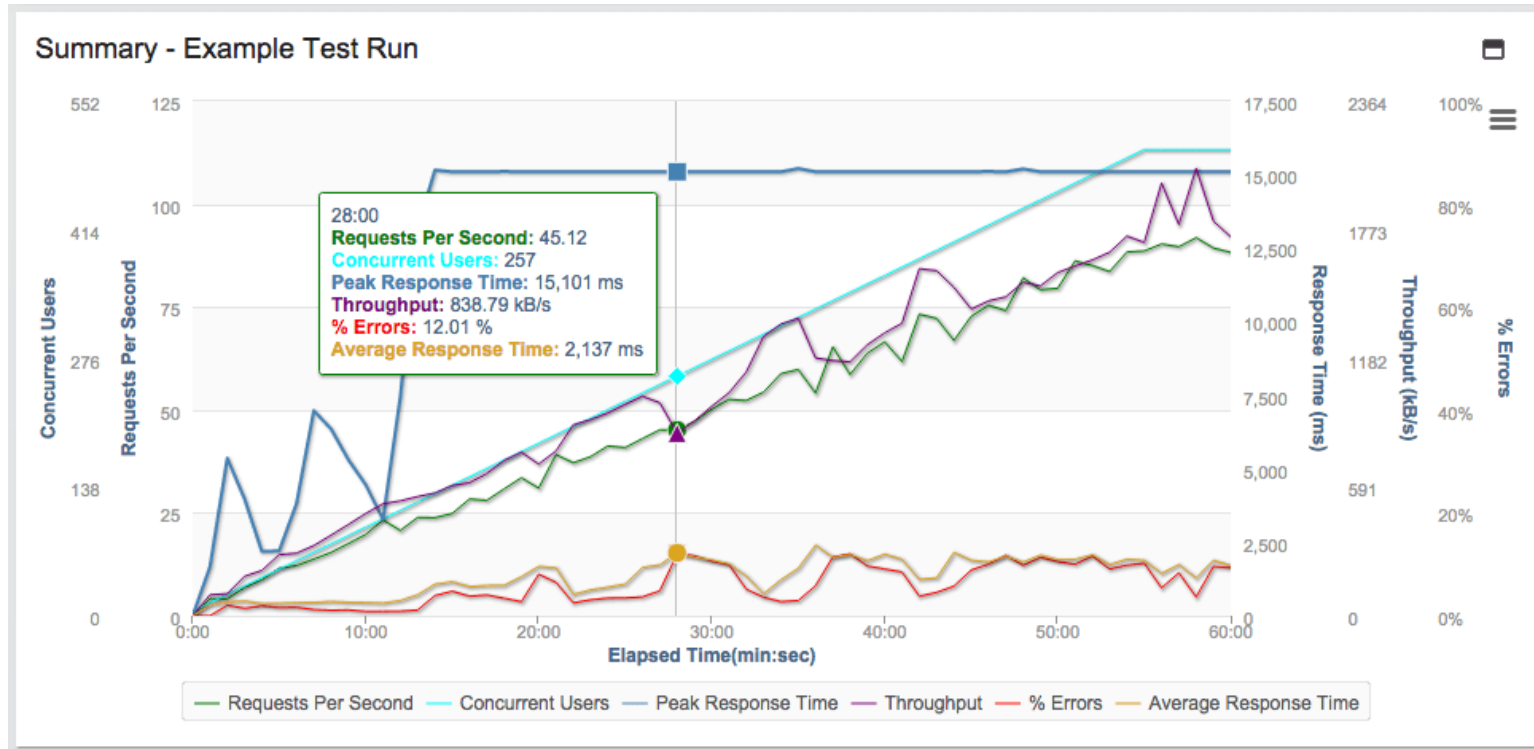
Aggregating performance testing results

Before aggregating results



Aggregating performance testing results

After aggregating results



Main concepts in performance testing

Outlines:

- Principles of Performance Testing.
- Load generation concept.
- Aggregating performance testing results.
- Common Performance Efficiency Failure Modes and their causes.
- Performance risks for different architecture.

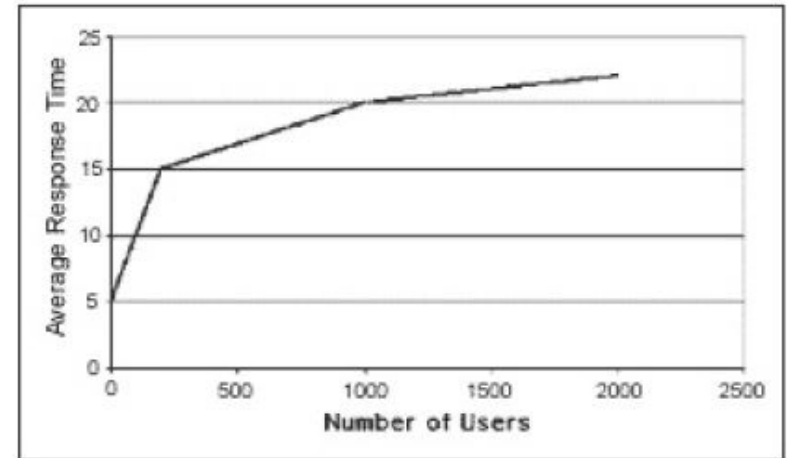


Common Performance Efficiency Failure Modes

➤ Slow response under all load levels:

In some cases, response is unacceptable regardless of load.

This may be caused by underlying performance issues, including, but not limited to, bad database design or implementation, network latency, and other background loads. Such issues can be identified during functional and usability testing, not just performance testing.



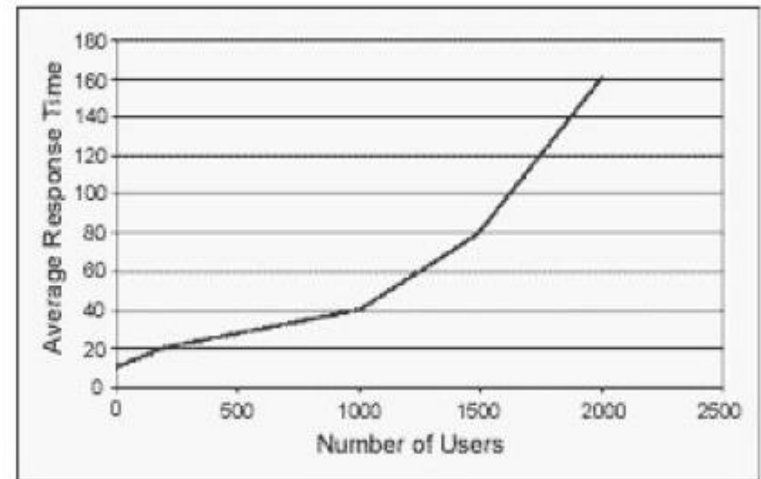
Common Performance Efficiency Failure Modes

➤ Slow response under moderate-to-heavy load levels:

In some cases, response degrades unacceptably with moderate-to-heavy load, even when such loads are entirely within normal, expected, allowed ranges.

There are a large number of technical elements which can impact the server response time, some of the main reasons are:

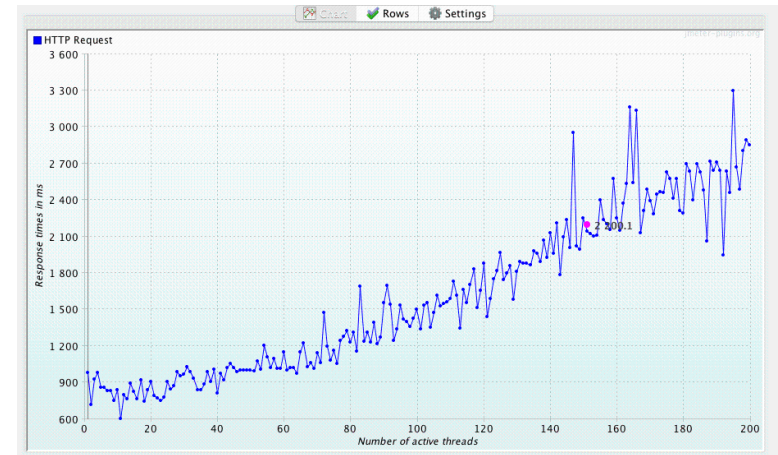
- Server specification
- Network bandwidth
- Website build quality
- Database queries
- Pages and data not being cached



Common Performance Efficiency Failure Modes

➤ Degraded response over time:

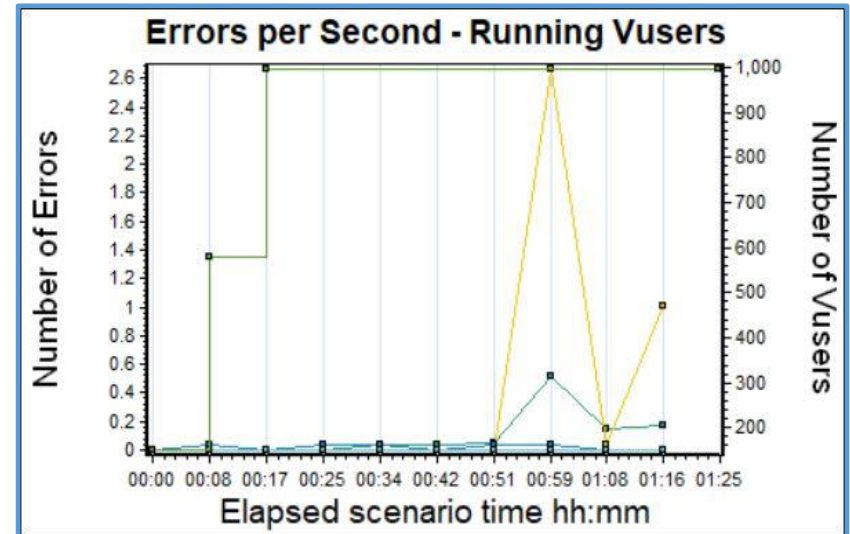
In some cases, response degrades gradually or severely over time. Underlying causes include memory leaks, disk fragmentation, increasing network load over time, growth of the file repository, and unexpected database growth.



Common Performance Efficiency Failure Modes

❖ Inadequate or graceless error handling under heavy or over-limit load:

In some cases, response time is acceptable but error handling degrades at high and beyond-limit load levels. Underlying defects include insufficient resource pools, undersized queues and stacks, and too rapid time-out settings.



Main concepts in performance testing

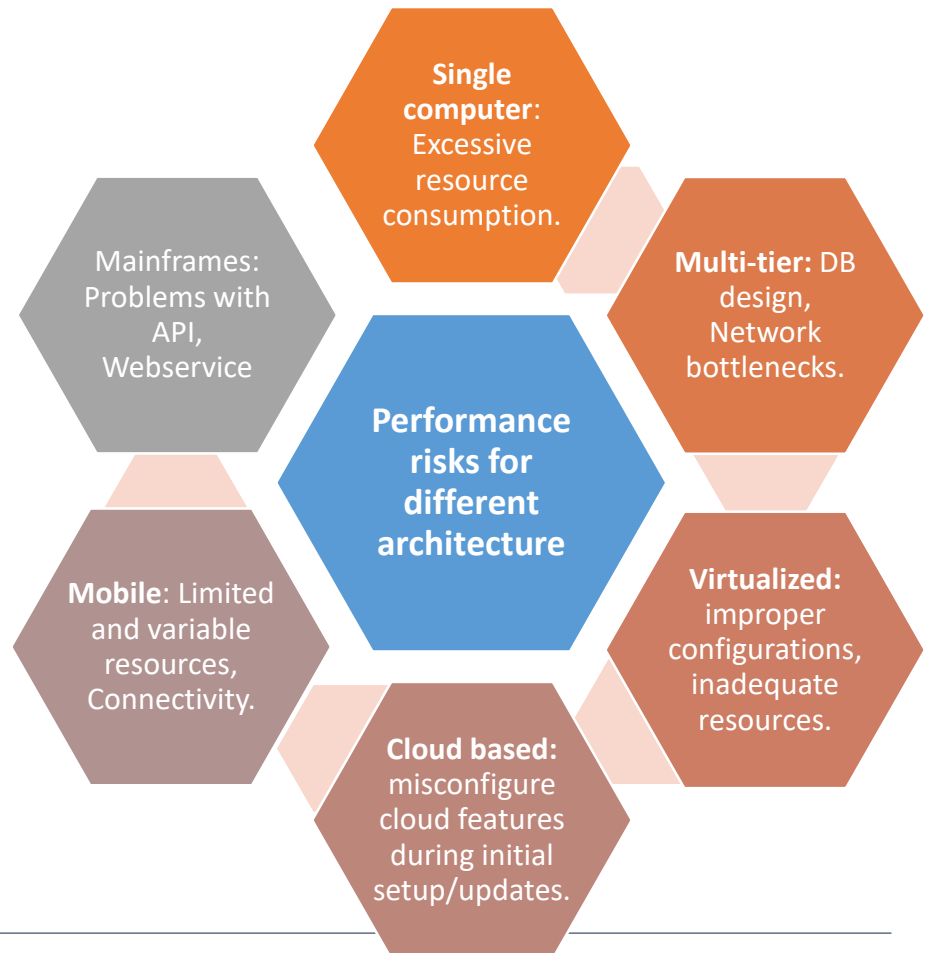
Outlines:

- Principles of Performance Testing.
- Load generation concept.
- Aggregating performance testing results.
- Common Performance Efficiency Failure Modes and their causes.
- Performance risks for different architecture.



Performance risks for different architecture

- It is very important to identify performance possible risks early in planning phase.
- Performance risks will differ based on the SUT architecture.
- Some performance risks are associated with some technical decisions: for example, memory leaks are most common with C and C++ programming languages!



Performance Testing Tools

Outlines:

- Tools Support
- Tools Suitability



Performance Testing Tools

Outlines:

- Tools Support
- Tools Suitability



Tools Support

➤ Load Generator:

The generator is able to create and execute multiple client instances that simulate user behavior according to a defined operational profile.

Creating multiple instances in short periods of time will cause **load** on a system under test. So, the load generators tools creates the load and also collects metrics for later reporting.

There are many load generator tools, and some of them listed below:

- ❖ Jmeter
- ❖ Taurus
- ❖ Element
- ❖ Locust
- ❖ Gatling
- ❖ K6

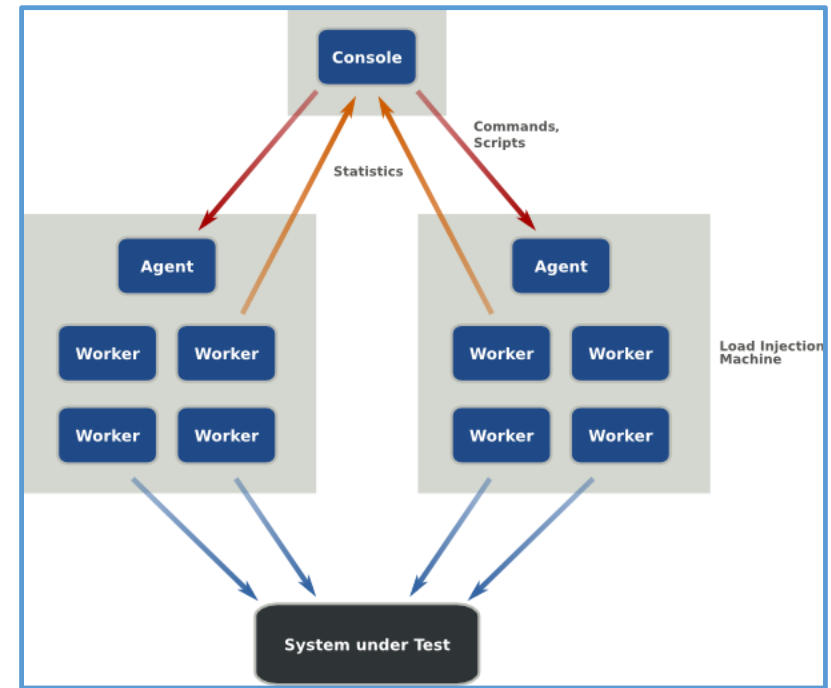


Tools Support

➤ Load Management Console:

The load management console provides the control to start and stop the load generator(s).

The console also aggregates metrics from the various transactions that are defined within the load instances used by the generator and enables reports and graphs from the test executions to be viewed and supports results analysis.



Tools Support

➤ Monitoring Tool:

Monitoring tools run concurrently with the component or system under test and supervise, record and/or analyze the behavior of the component or system.

Monitoring tools can effectively support the root cause analysis of performance degradation in a system under test and may also be used to monitor a production environment when the product is released.

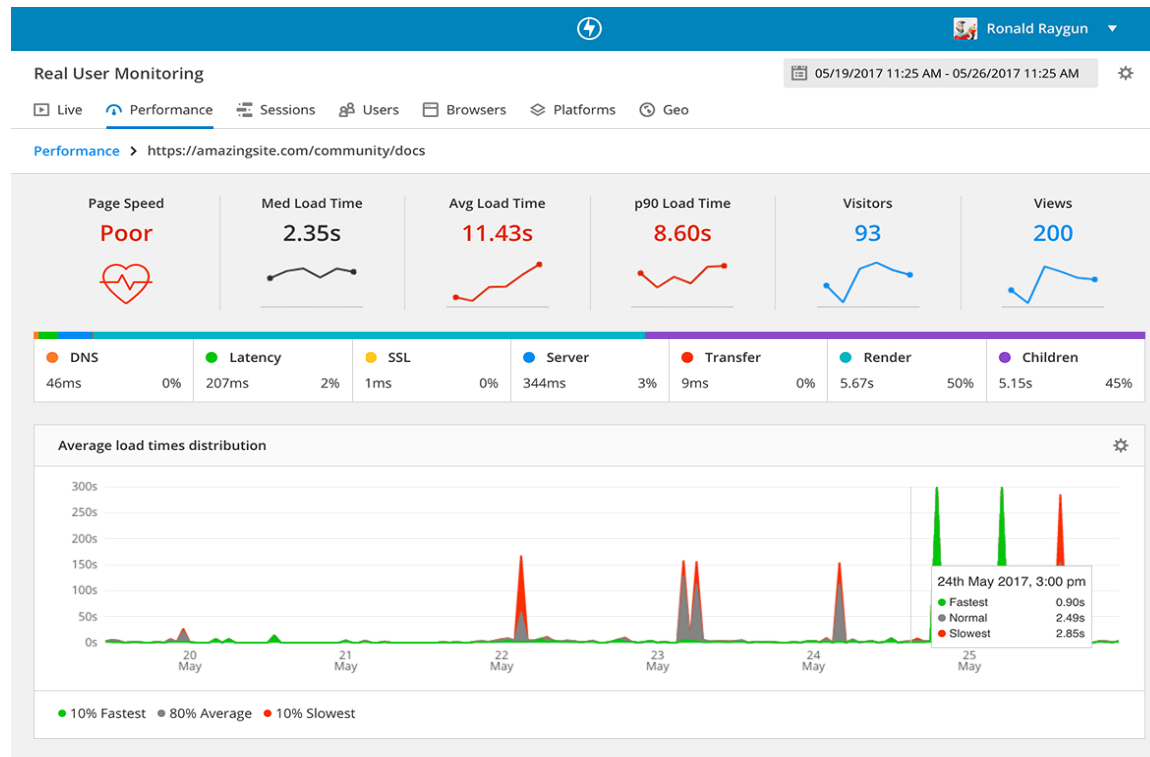
Typical components which are monitored include web server queues, system memory and disk space.



Tools Support

Monitoring Tool Examples:

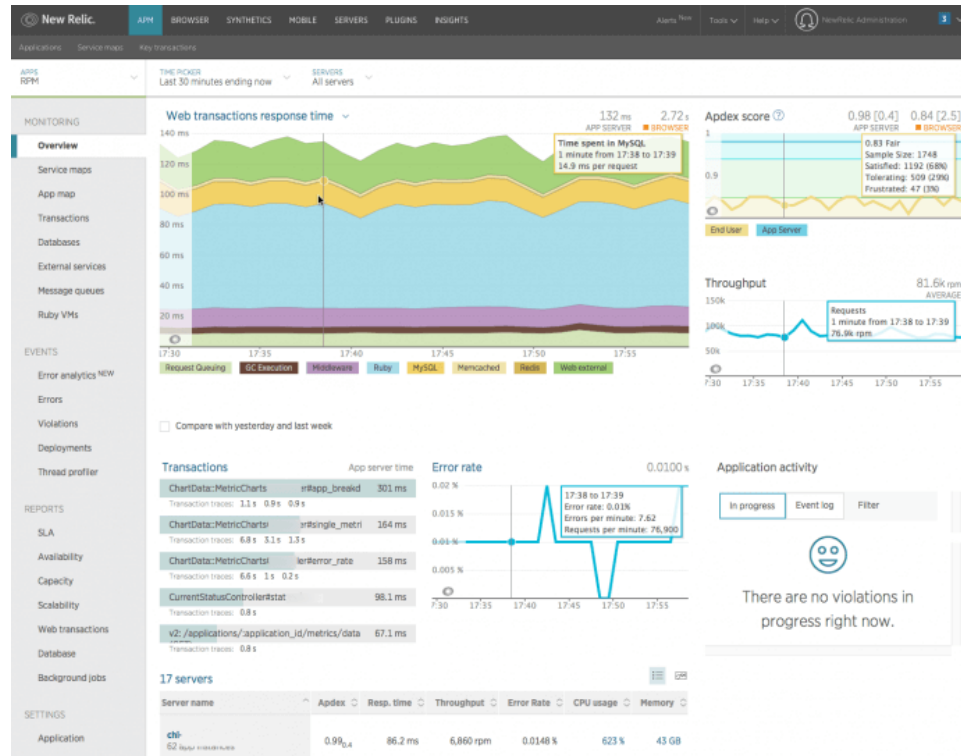
➤ Raygun



Tools Support

Monitoring Tool Examples:

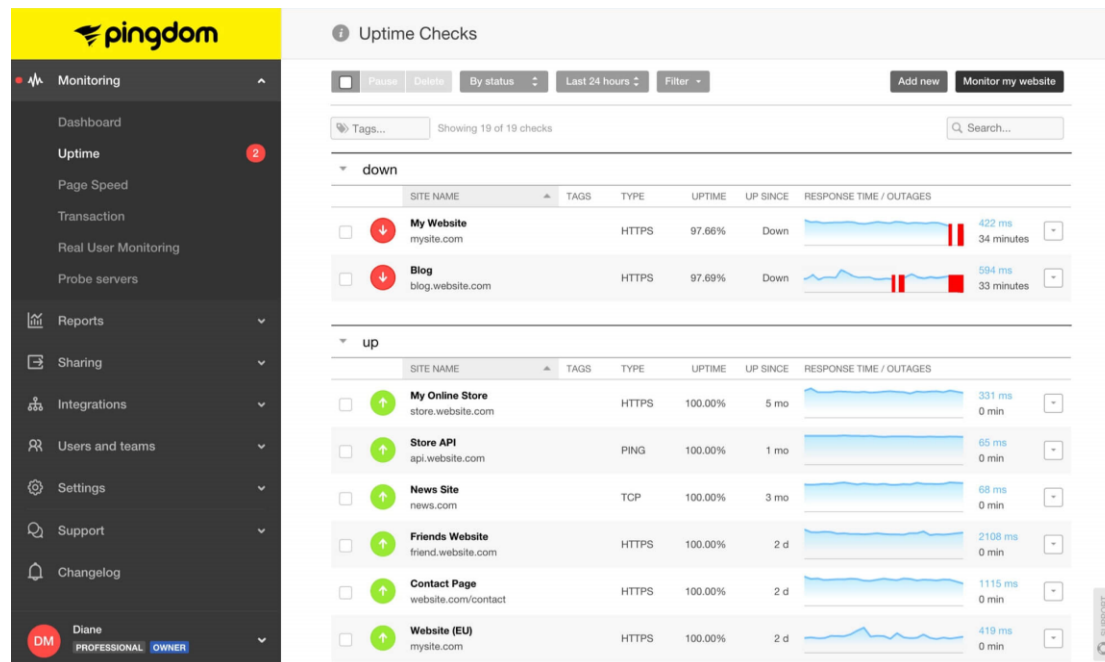
➤ New Relic



Tools Support

Monitoring Tool Examples:

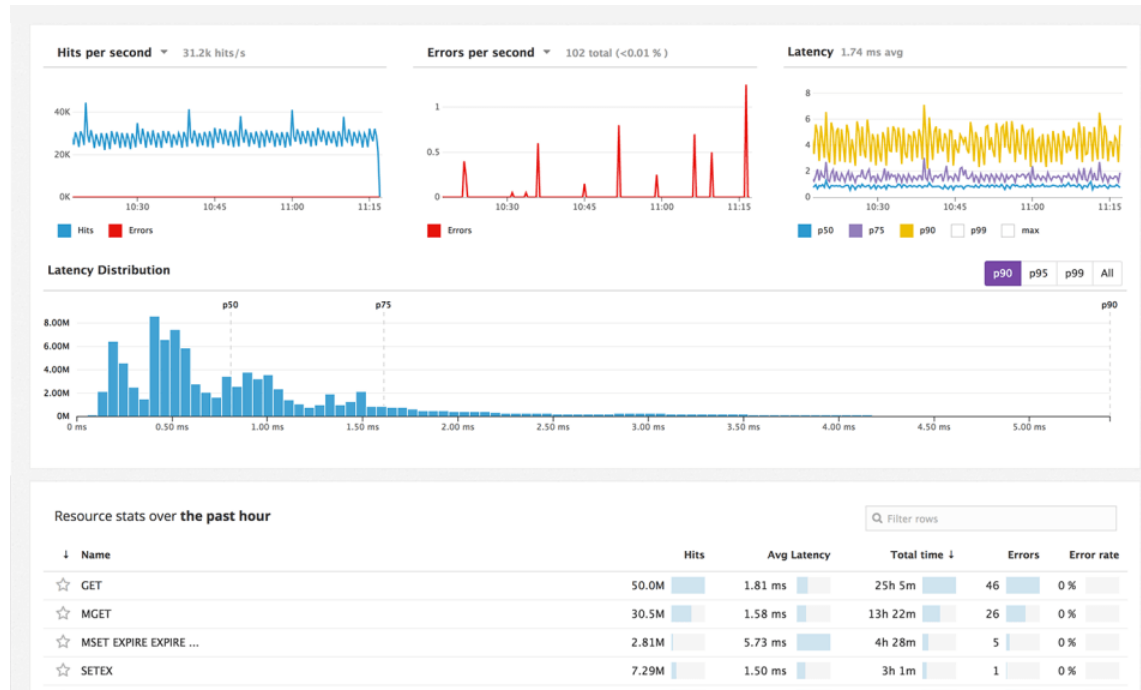
➤ Pingdom



Tools Support

Monitoring Tool Examples:

➤ Datadog



Performance Testing Tools

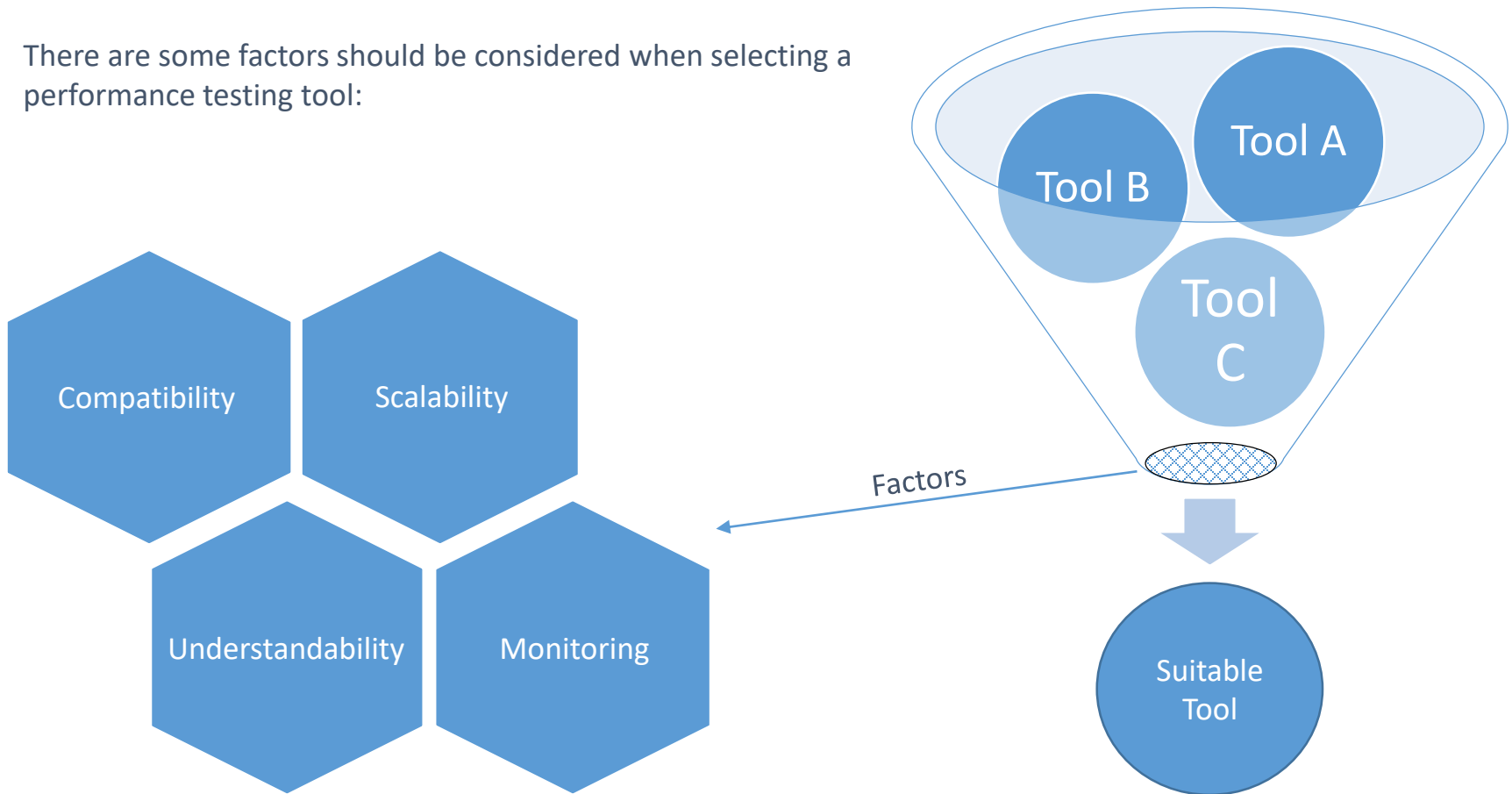
Outlines:

- Tools Support
- Tools Suitability



Tool Suitability

There are some factors should be considered when selecting a performance testing tool:



Tool Suitability

➤ **Compatibility:**

We should consider the following factors:

- **Protocols:** Understanding which protocols a system uses and which of these will be tested will help to evaluate the appropriate test tool.
- **Interfaces to external components:** This factor may need to be considered as part of the complete integration requirements to meet process or other inter-operability requirements.
- **Platforms:** Compatibility with the platforms (and their versions) within an organization is essential. This applies to the platforms used to host the tools and the platforms with which the tools interact for monitoring and/or load generation.



Tool Suitability

➤ **Scalability:**

This will include several factors:

- Maximum number of licenses required.
- Load generation workstation/server configuration requirements.
- Ability to generate load from multiple points of presence (e.g., distributed servers)



Tool Suitability

➤ Understandability:

Is the level of technical knowledge needed to use the tool.

Without considering this factor, this can lead to unskilled testers incorrectly configuring tests, which in turn provide inaccurate results.

For testing requiring complex scenarios and a high level of programmability and customization, teams should ensure that the tester has the necessary skills, background, and training.



Tool Suitability

➤ **Monitoring:**

There are some questions must be answered to determine if the tool will provide the monitoring required by the project:

- Is the monitoring provided by the tool sufficient?
- Are there other monitoring tools available in the environment that can be used to supplement the monitoring by the tool?
- Can the monitoring be correlated to the defined transactions?



Performance testing in Software Lifecycle

Outlines:

- Principal performance testing activities and SDLC
- Planning.
- Analysis, design and implementation.
- Execution.
- Analysing results and reporting.



Performance testing in Software Lifecycle

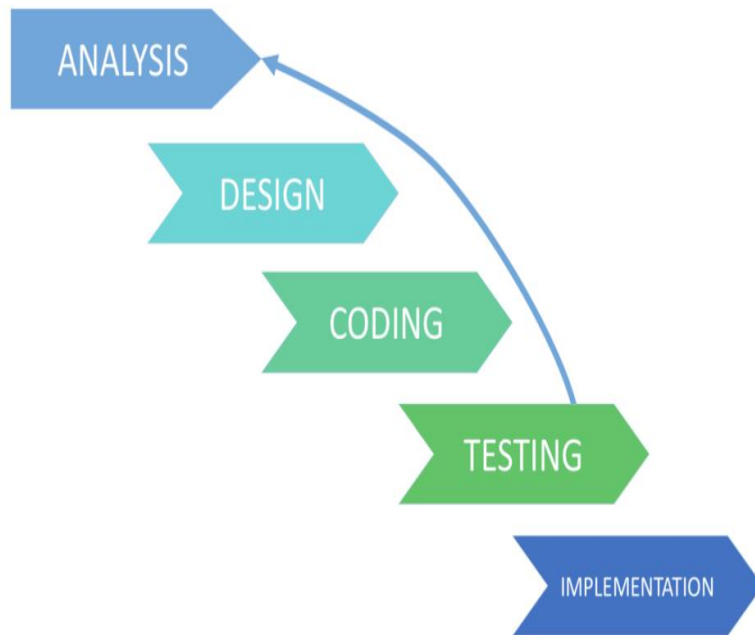
Outlines:

- Principal performance testing activities and SDLC
- Planning.
- Analysis, design and implementation.
- Execution.
- Analysing results and reporting.

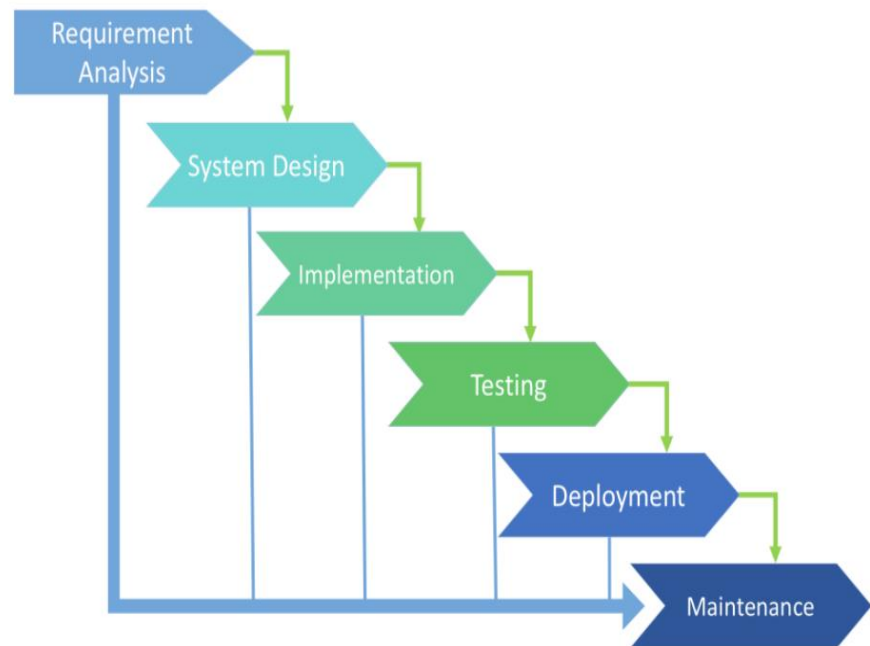


Performance testing activities will be part of each stage based on the followed SDLC:

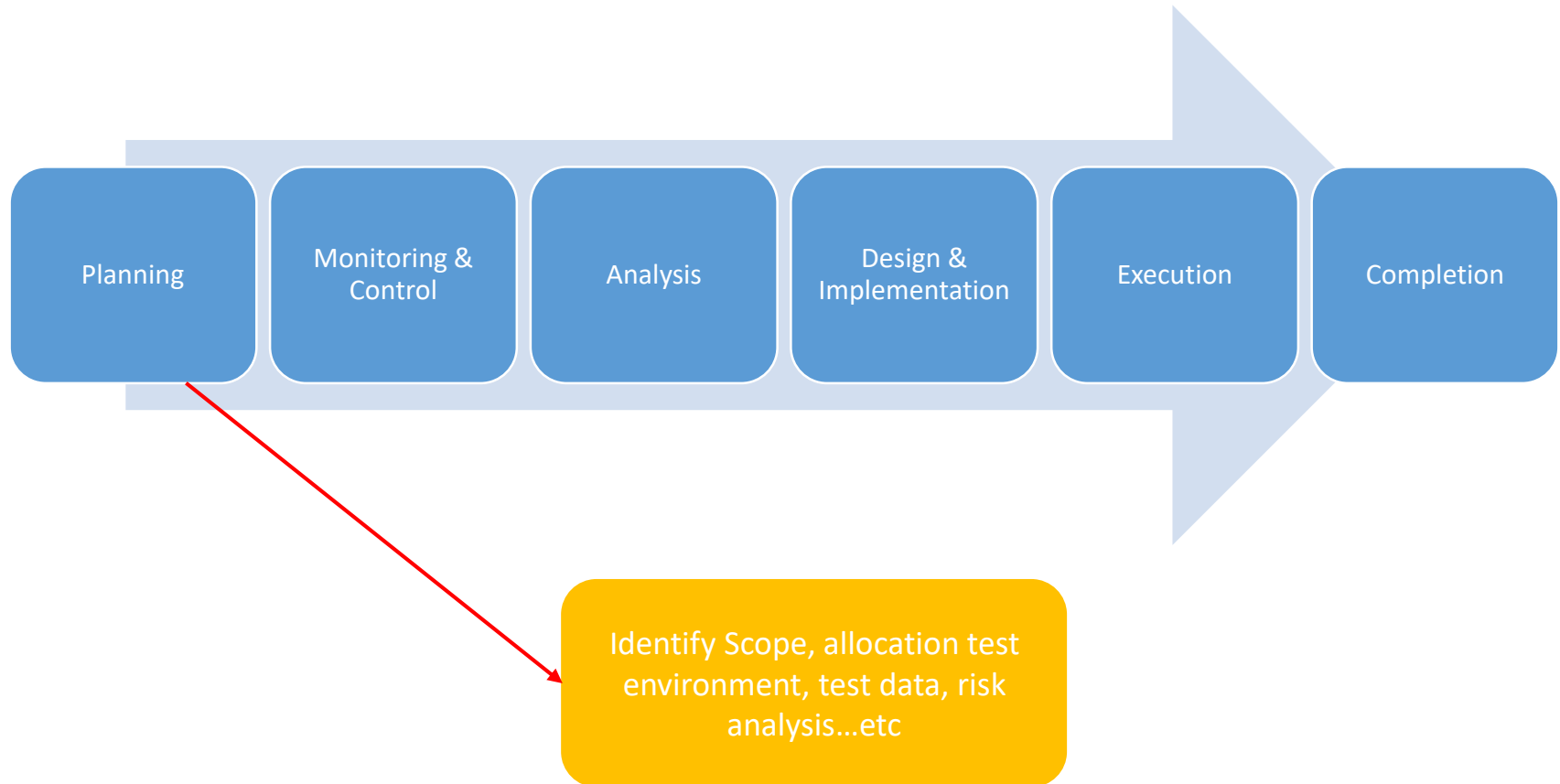
Iterative Model:



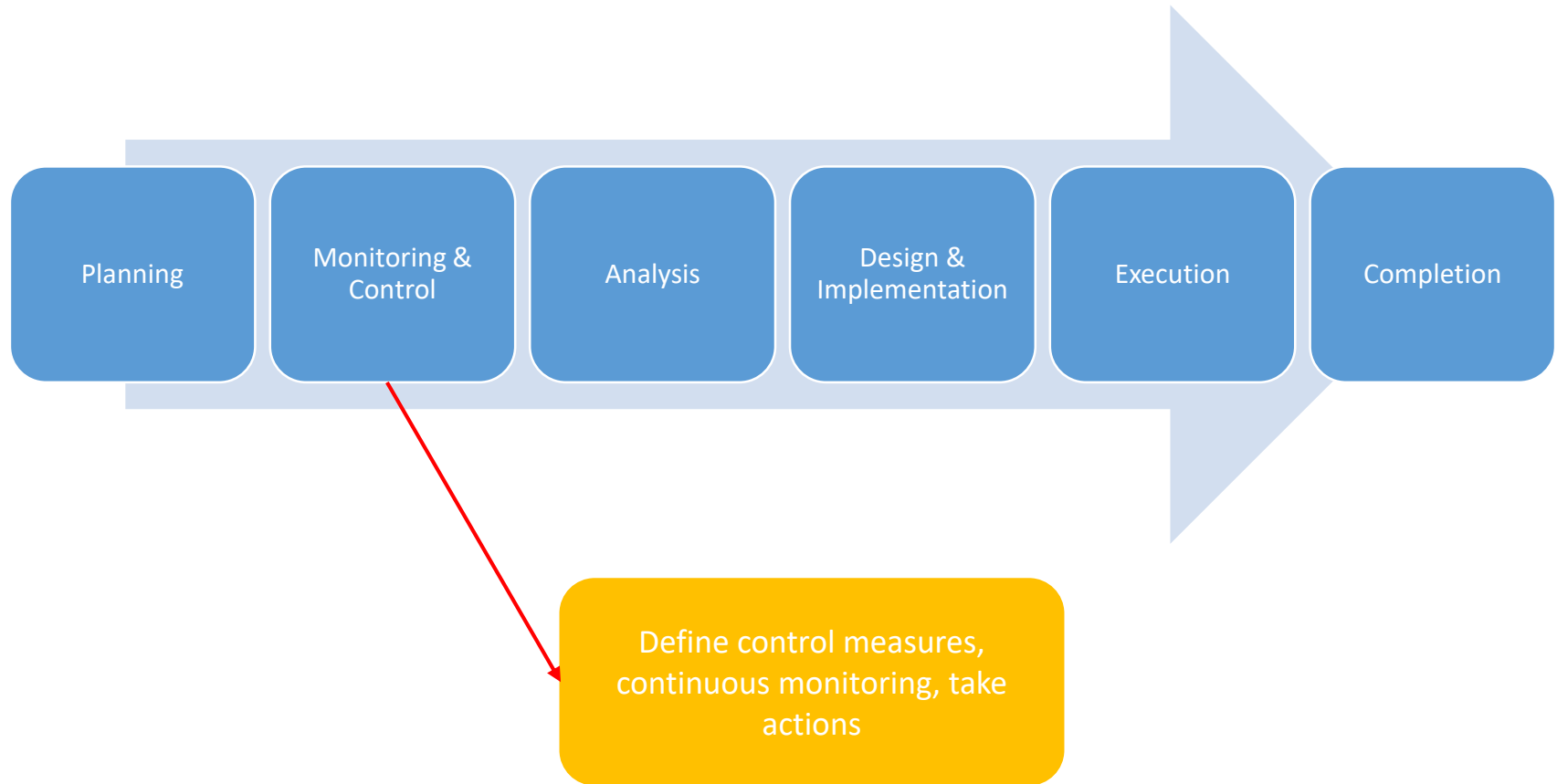
Sequential model:



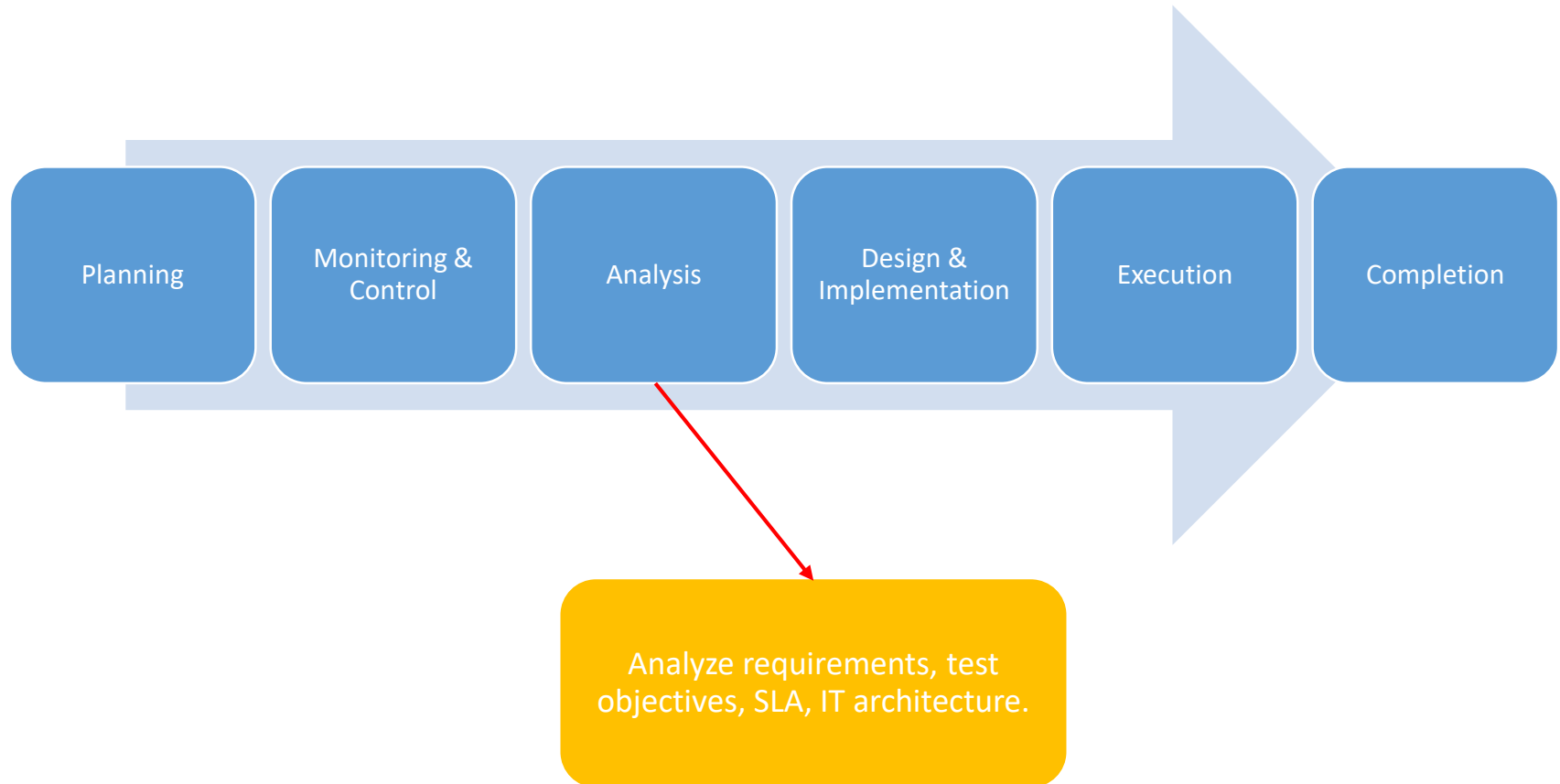
Principal performance testing activities



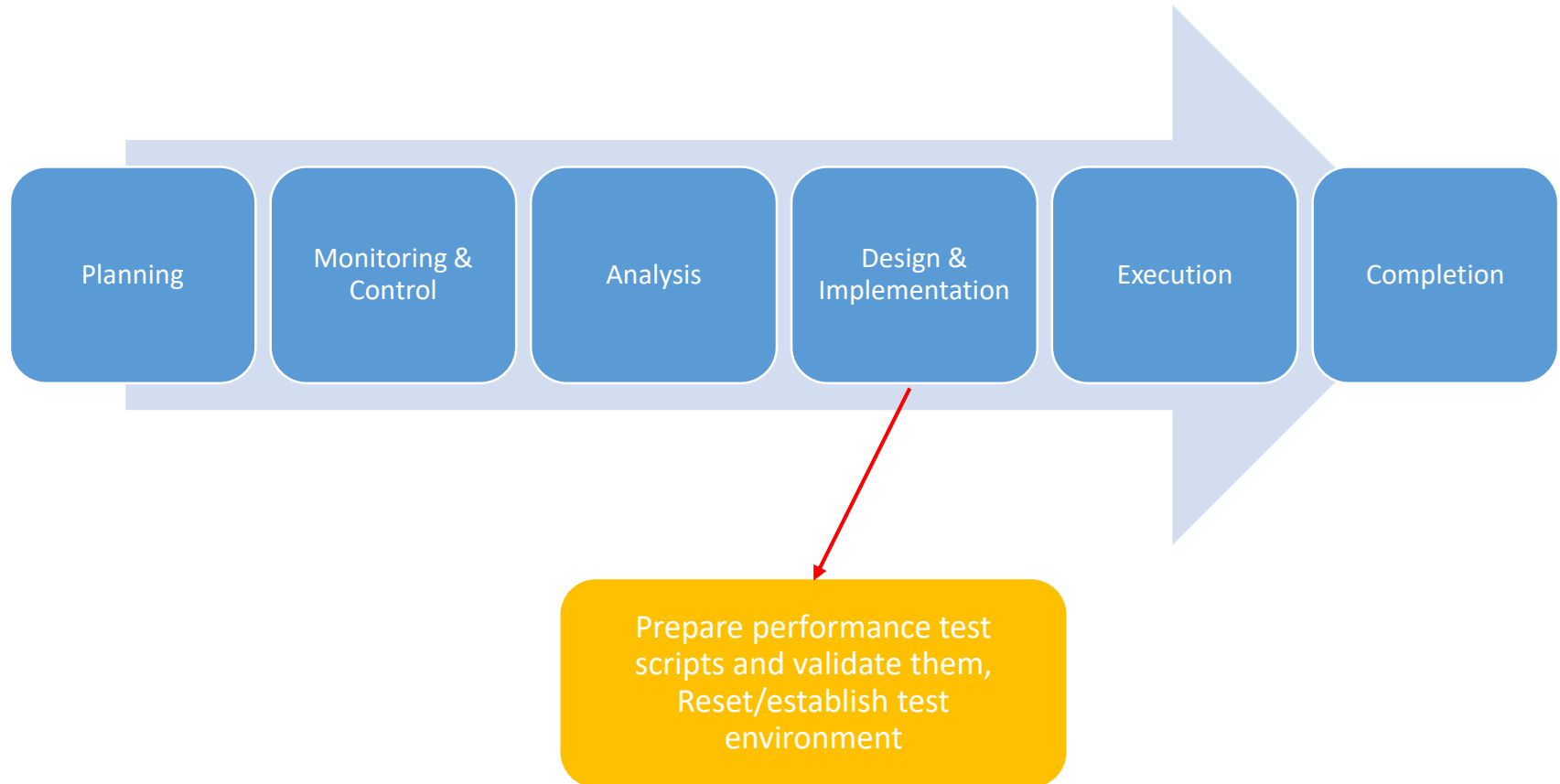
Principal performance testing activities



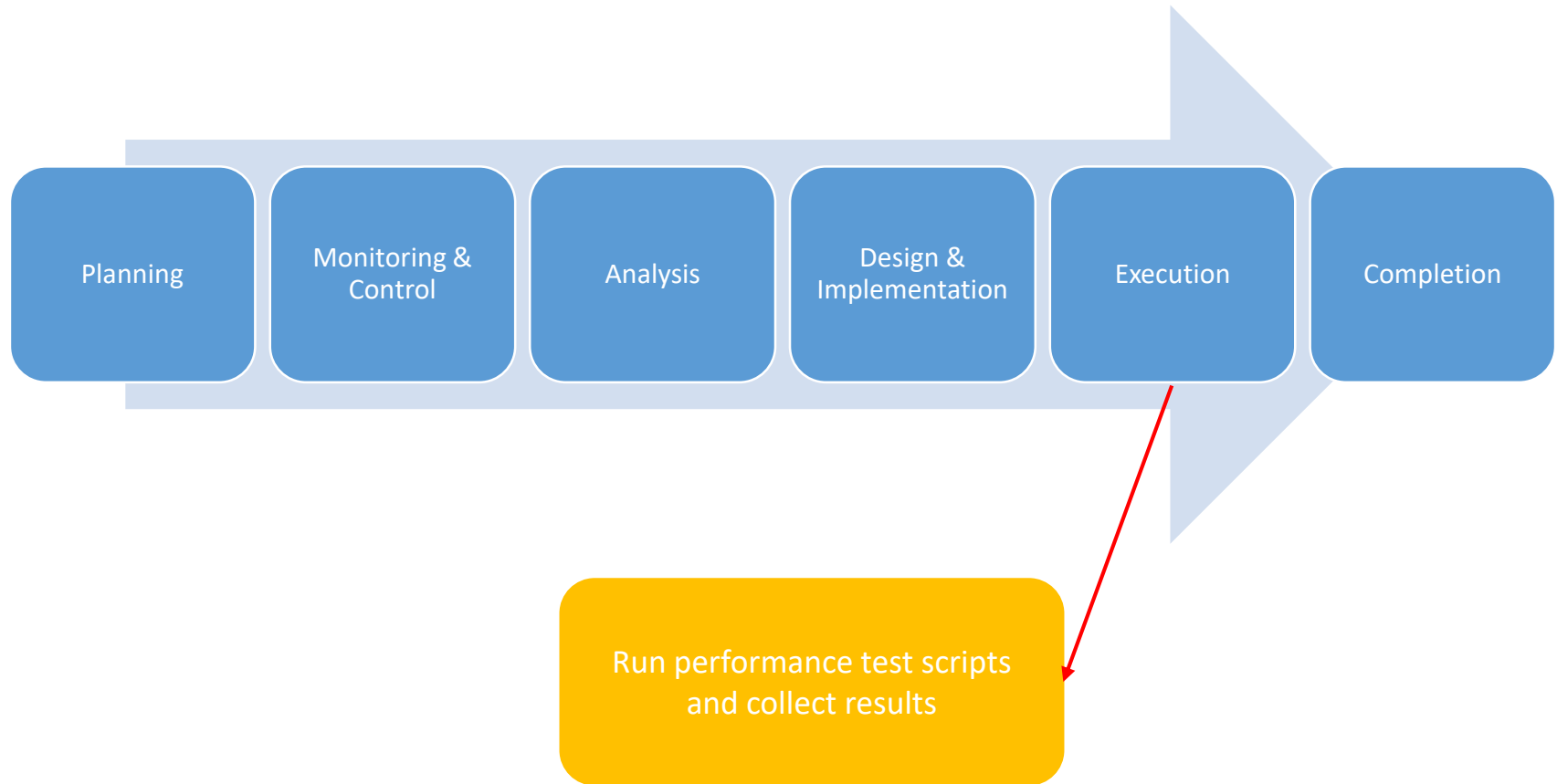
Principal performance testing activities



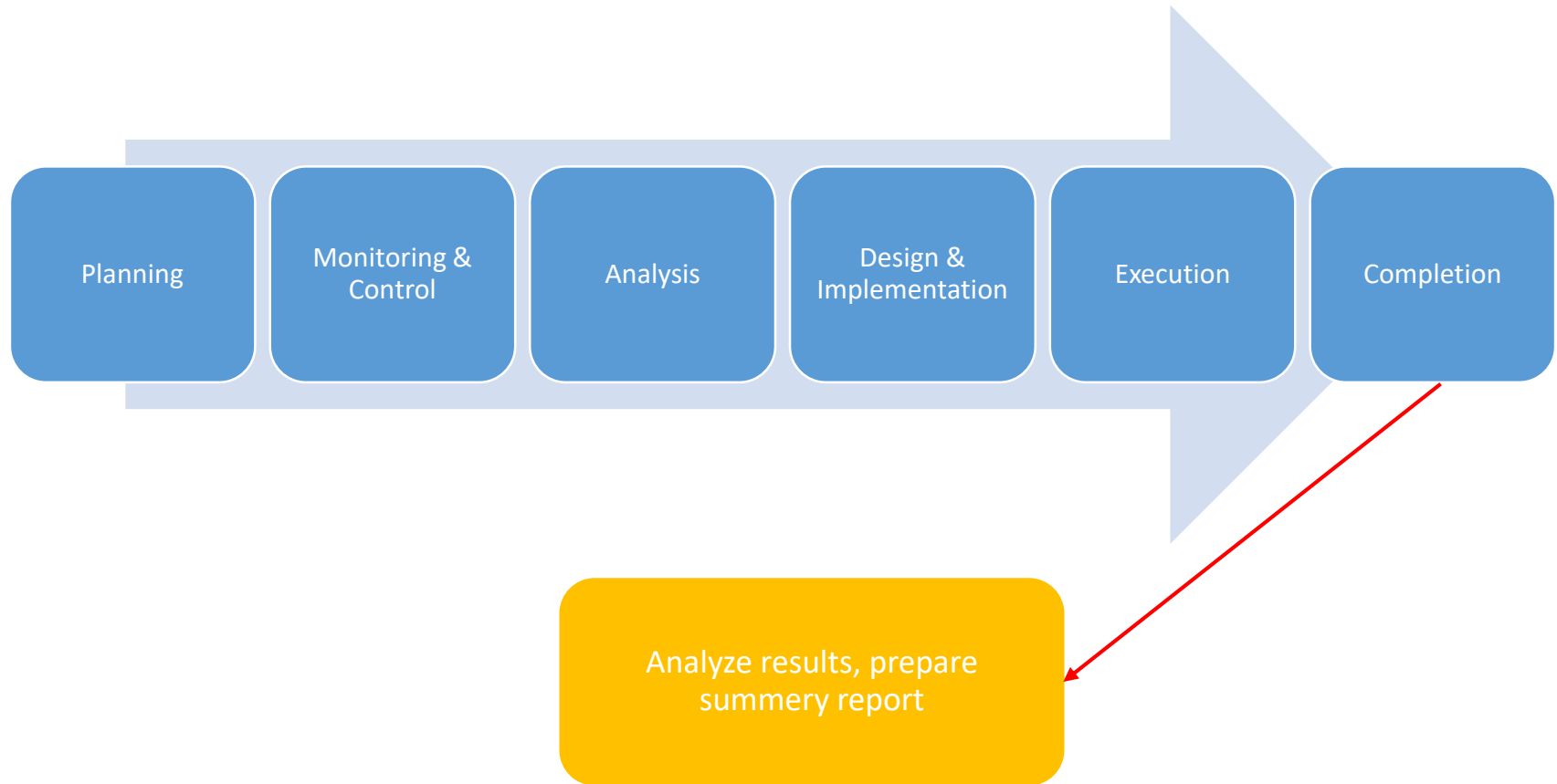
Principal performance testing activities



Principal performance testing activities



Principal performance testing activities



Performance testing in Software Lifecycle

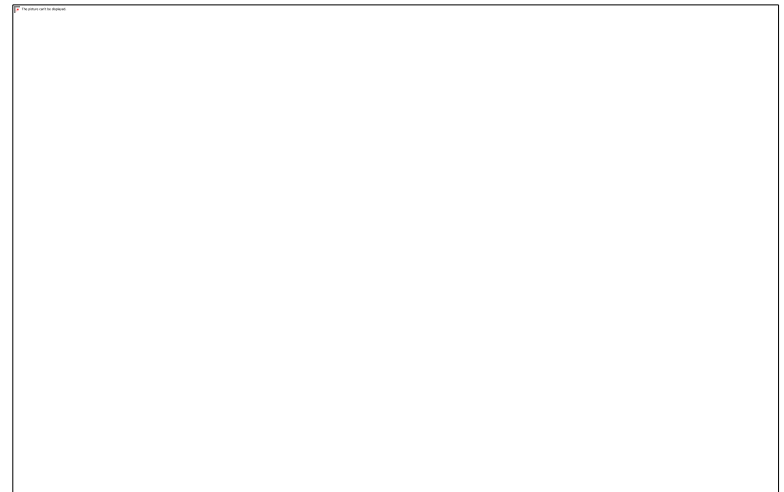
Outlines:

- Principal performance testing activities and SDLC
- Planning.
- Analysis, design and implementation.
- Execution.
- Analysing results and reporting.



Planning – Activates

- Drive performance test objectives from relevant information:
 - What transactions will be executed?
 - What average response time is expected?
 - What system metrics will be captured? And what values are expected?
- Identify possible risks.
- Prepare Performance test plan.



Planning – Test plan

Performance test plan should includes the below:

- **Objective:** describe the goals and strategies for performance testing (Why to test?)
- **System overview:** A brief description of the SUT, architecture, specs..etc (What to test?)
- **Types of performance testing to be conducted:** needed types along with a description and the purpose of each type (How to test?)
- **Acceptance criteria:** define the conditions that the SUT must meet to be accepted by a user, a customer, or other systems (How to test?)
- **Test data:** define the needed test data, its types, who will provide it and how it will be used in testing (How to test?)
- **Resources tools:** define the human resources that will participate in testing activates and the tools that will be used fore scripting, executing and monitoring. (who/where/How to test?)
- **Operational and load profiles/ test scenarios:** define the repeatable step by step flow through the application for a particular usage of the system with its expected load level. (When/How to test?)
- **Risks:** Identify project and product possible risks and how to mitigate it. (How to test?)



Planning – Practice

Let's practice!

Scenario:

You are working for a company that has developed a web application that will be used in hospital management. This application will be used by hospital managers, doctors and nurses as below (they will log in to the system using unique user name and password):

Managers: 4 managers, will view all reports and will use the system concurrently from 9am-4pm to view nearly 2 reports per hour.

Doctors: 35 doctor, will Input patient details, view some reports and approve some forms. will use the system concurrently 24h/day and each doctor will perform 5 transactions per hour.

Nurses: 70 nurse, Input patient details, view some reports. will use the system concurrently 24h/day and each nurse will perform 10 transactions per hour.

The overall system consists of DB server, Application server, reports server and client server (users PC).

Response time should not exceed 4 seconds in all pages and resources should not exceed 80% at any situation.

We need to measure system capabilities under real life expected loads and evaluate system behavior under heavy load (beyond the expected)



Planning – Practice

Let's practice!

Scenario:

You are working for a company that has developed a web application that will be used in hospital management. This application will be used by hospital managers, doctors and nurses as below (they will log in to the system using unique user name and password):

Managers: 4 managers, will view all reports and will use the system concurrently from 9am-4pm to view nearly 2 reports per hour.

Doctors: 35 doctor, will Input patient details, view some reports and approve some forms. will use the system concurrently 24h/day and each doctor will perform 5 transactions per hour.

Nurses: 70 nurse, Input patient details, view some reports. will use the system concurrently 24h/day and each nurse will perform 10 transactions per hour.

The overall system consists of DB server, Application server, reports server and client server (users PC).

Response time should not exceed 4 seconds in all pages and resources should not exceed 80% at any situation.

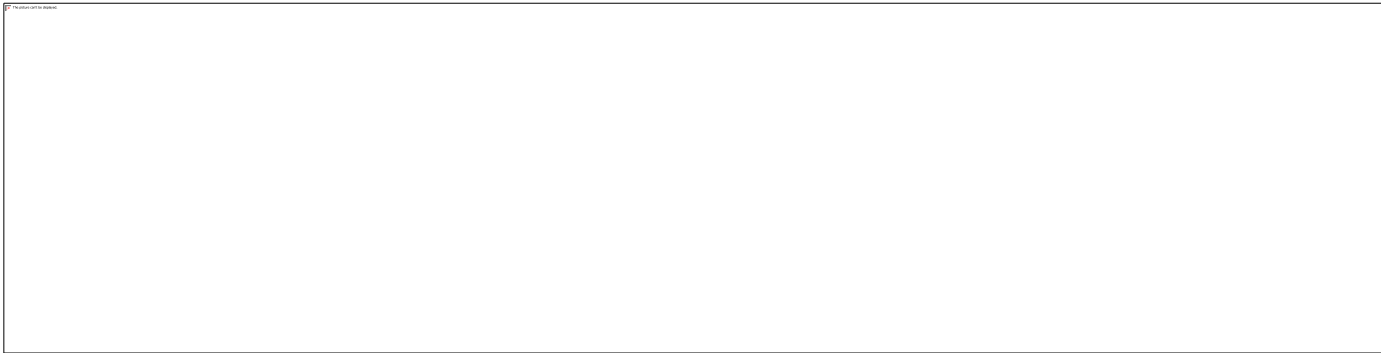
We need to measure system capabilities under real life expected loads and evaluate system behavior under heavy load (beyond the expected)



Planning – Practice

From the mentioned scenario, define:

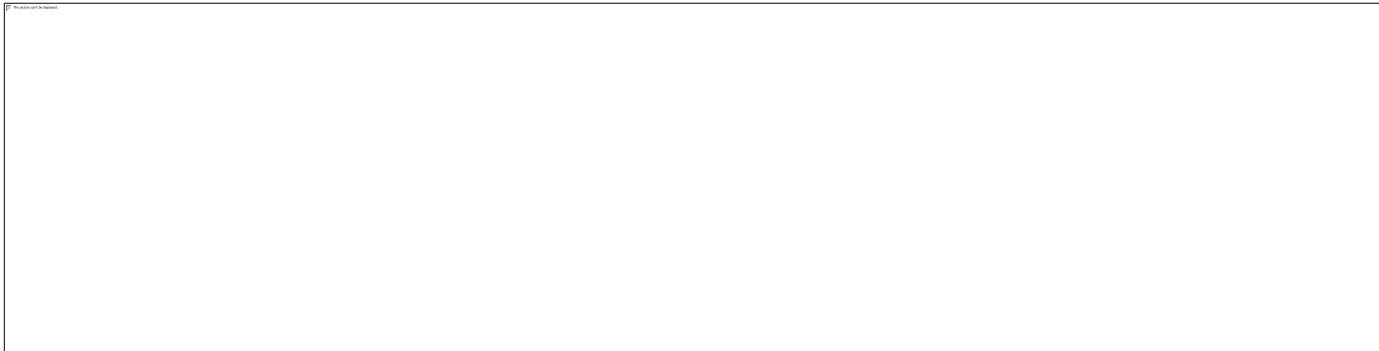
- Test objective
- System overview
- Test types
- Acceptance criteria
- Test operational and load profiles
- Risks regarding the architecture



Planning – Practice

From the mentioned scenario:

- **Test objective** : to measure system capabilities under real life expected loads and evaluate system behavior under heavy load (beyond the expected)
- **System overview**: DB server, Application server, reports server and client server (users PC).
- **Test types** : load and stress test.
- **Acceptance criteria** :Response time should not exceed 4 seconds in all pages and resources should not exceed 80% at any situation.



Planning – Practice

From the mentioned scenario:

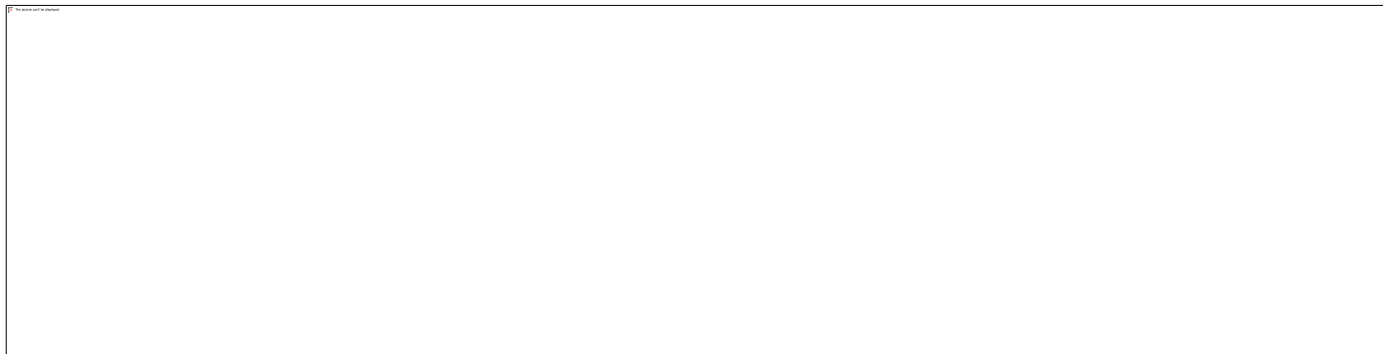
➤ Test operational and load profiles:

Managers: 4 managers, will view all reports and will use the system concurrently from 9am-4pm to view nearly 10 reports per user.

Doctors: 35 doctor, will Input patient details, view some reports and approve some forms. will use the system concurrently 24h/day and each doctor will perform 3-5 transactions per hour.

Nurses: 70 nurse, Input patient details, view some reports. will use the system concurrently 24h/day and each nurse will perform 3-5 transactions per hour.

➤ Risks regarding the architecture : Multi-tier (DB design, Network bottlenecks).



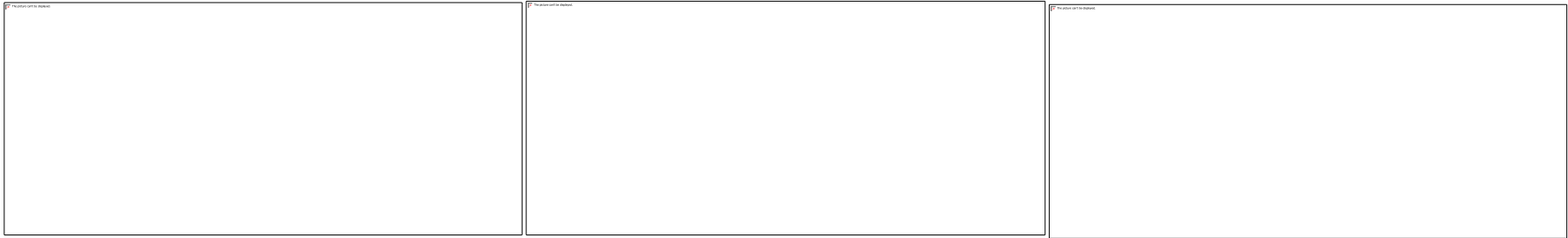
Performance testing in Software Lifecycle

Outlines:

- Principal performance testing activities and SDLC
- Planning.
- Analysis, design and implementation.
- Execution.
- Analysing results and reporting.



Analysis, design and implementation.



- Construct Operational profile.
- Construct Load profiles (Ramp-ups , Ramp-downs, predefined distributions).
- Identify and gather test data (Different types of users and their roles, tasks performed by the users, estimated number of users per unit of time.)
- Design test scripts (Protocol level, GUI level) in a way that it contains : initialization section, main section, clean up sectional and some times results verification and error handling section.
- Validate the designed scripts.
- Setting up the system under test.
- Deploying the environment.
- Setting up the load generation and monitoring tools.





الْجَمْعِيَّةُ الْعِلْمِيَّةُ الْمَلَكِيَّةُ
Royal Scientific Society



الْجَمْعِيَّةُ الْعِلْمِيَّةُ الْمَلَكِيَّةُ
Royal Scientific Society



الْجَمْعِيَّةُ الْعِلْمِيَّةُ الْمَلَكِيَّةُ
Royal Scientific Society

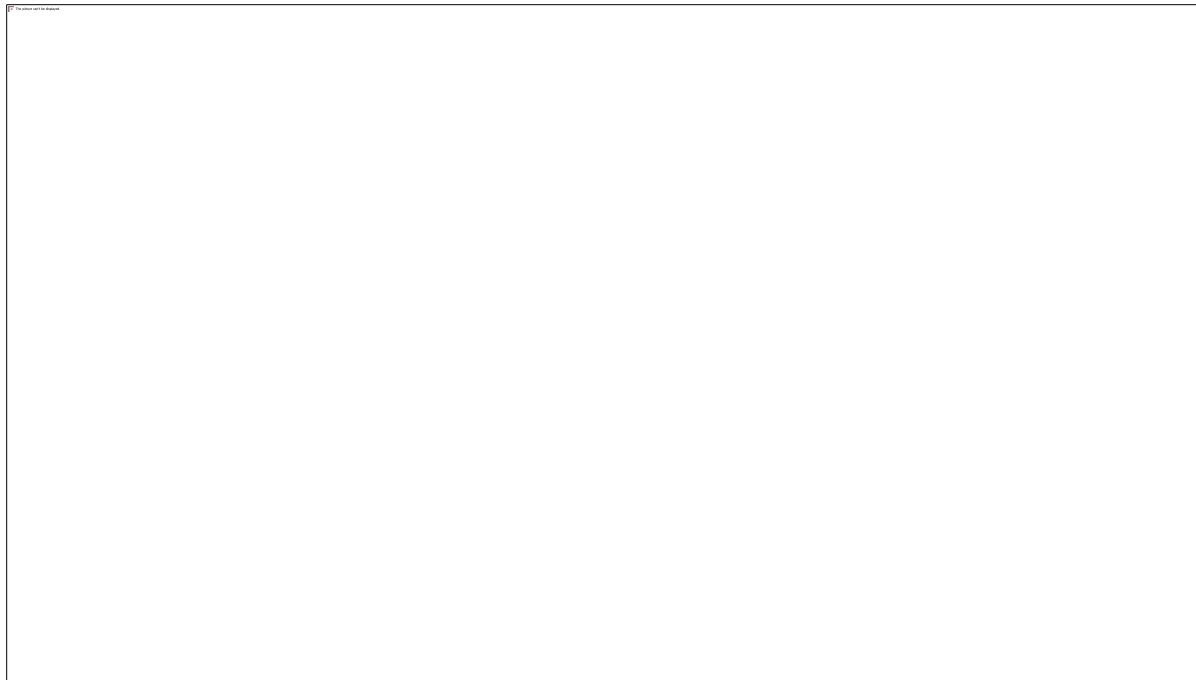


الْجَمْعِيَّةُ الْعِلْمِيَّةُ الْمَلَكِيَّةُ
Royal Scientific Society

Analysis - Practice

Lets practice!

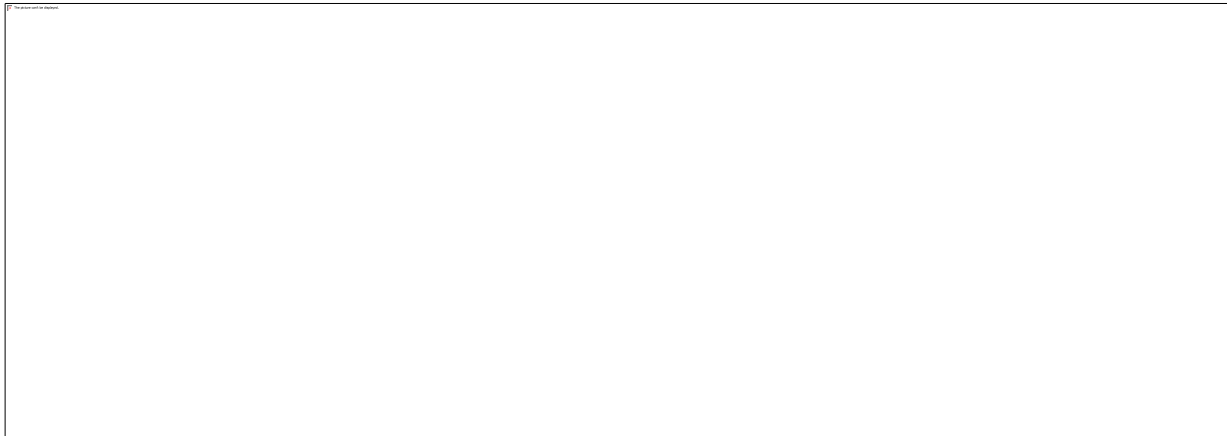
Below is an example for a performance test scenario for a customer service role for “X” web application:



Analysis - Practice

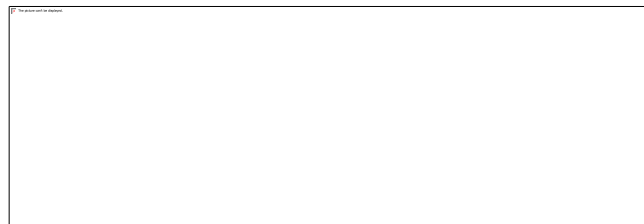
Lets practice!

10 concurrent customer services users will use the “X” web application for 8 hours per day, daily average number of received requests by all customer service employees are 160, below is how can we design the ramp-up and ramp-downs:



Design

- After the analysis, performance tester starts developing the performance test scripts according to the predefined test scenarios.
- Scripts can be developed:
 - **Recording (GUI or protocol level)**
 - **Programming**
- Performance script is one or several sections of code that will include :
 - **Server requests creating load**
 - **Some programming around them specifying how exactly these requests would be invoked (in what order, at what moment, with what parameters, what should be checked...etc)**
- One of the tools used for performance scripting is JMeter.
- Each script needs to be validated before execution.



Implementation

Main activities in implementation phase can be summarized as below:

- **Deploying the environment:** the environment should be close to the production environment as much as possible.
- **Setting up the system under test:** data (size and structure), HW/SW/network configurations
- **Setting up the load generation and monitoring tools:** to make sure that all necessary information will be collected.

Now we are ready for test execution ...



Performance testing in Software Lifecycle

Outlines:

- Principal performance testing activities and SDLC
- Planning.
- Analysis, design and implementation.
- Execution.
- Analysing results and reporting.



Execution

- Execution is when we:
 - Generate load against the SUT according to a load profile.
 - Monitoring all parts of the environment and collect results.
- Test execution will include 3 stages:
 - **Ramp up:** the stage of getting to the steady state (using incremental load states).
 - **Steady state:** when all simulated users are initiated and are performing work as designed.
 - **Ramp down:** the state of finishing the test.
- Performance test are usually focused on steady state of the system. When the load is changing the system behavior is changing and it becomes more difficult to monitor and analyze test results
- Also it is important to test transient states when system behavior is changing (we need to carefully monitor and analyze results as monitoring averages will be misleading)



Performance testing in Software Lifecycle

Outlines:

- Principal performance testing activities and SDLC
- Planning.
- Analysis, design and implementation.
- Execution.
- Analysing results and reporting.



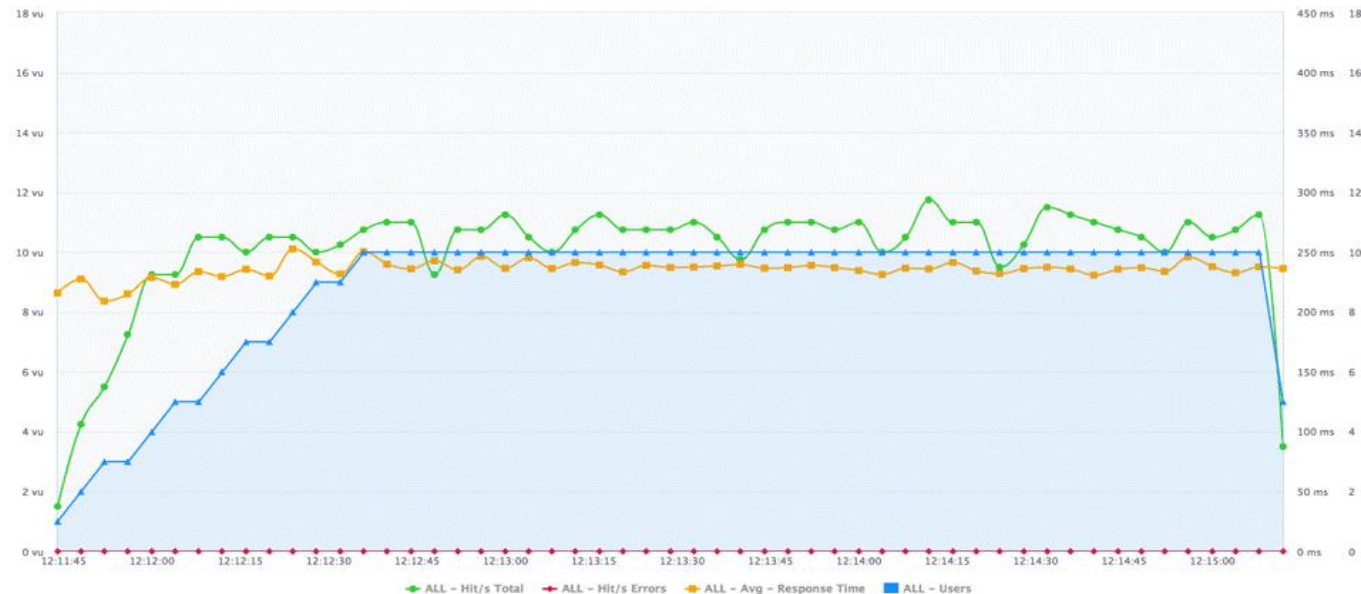
Analysing results and reporting

- When analyzing the results we first compare it to the performance objectives and acceptance criteria.
- Once the behavior understood, conclusions/recommendations can be drawn (changing physical components/SW/network configurations).
- We usually analyze:
 - Status of simulated users.
 - Transaction response time.
 - Transaction per second.
 - Transaction failures.
 - Hits per second.
 - Network throughput.
 - HTTP responses.



Analysing results and reporting

Lets practice!

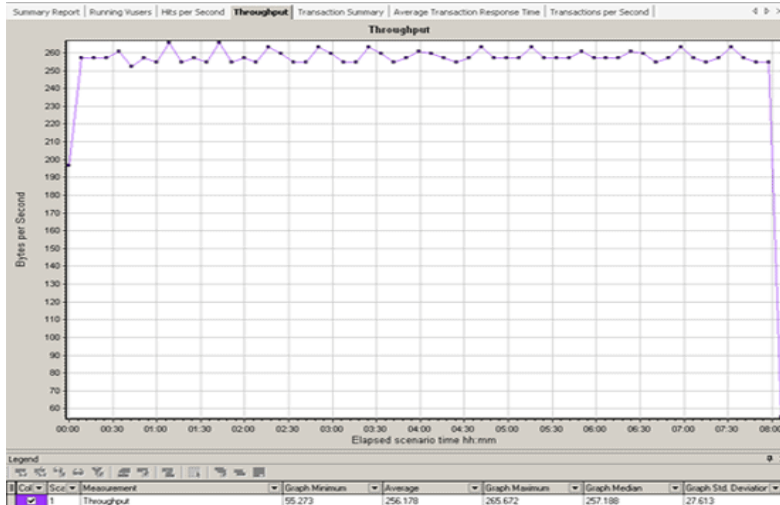


- This is a good result for a number of reasons. We want to see the blue and green lines climbing while the yellow one stays relatively flat. If the yellow line drops, we know that the software can't stand up to high demands (users and visitors).
- As we add hits to the software, the system should be able to withstand traffic until we reach our expected weight limit. Seeing a drop in the yellow line on the graph would indicate that there is a problem and that the weight limit is a lot lower than desired.

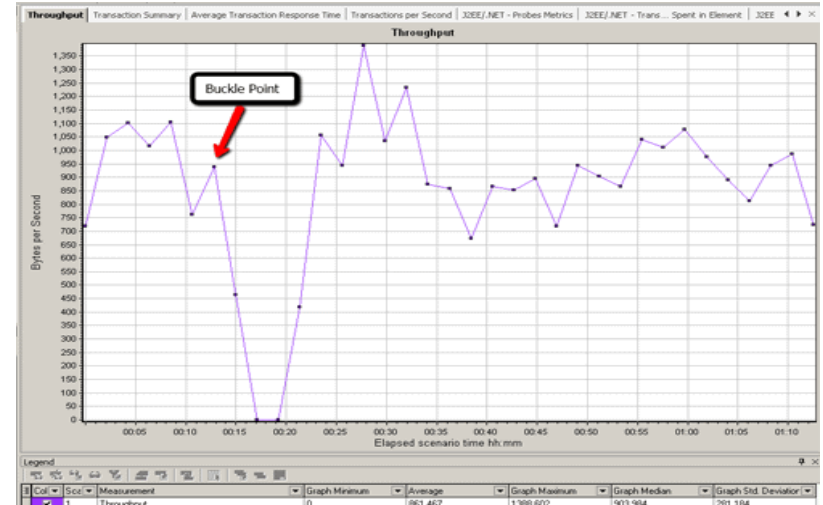


Analysing results and reporting

Lets practice!



Very good results for throughput.



Bad results for throughput.



Analysing results and reporting

- After we are done with analyzing results we need to prepare test summary report, it should contain:
- Executive summary
 - Aggregated Test results
 - Recommendations and conclusions



Practical Work

Outlines:

- JMeter prerequisites and installation.
- Main JMeter components.
- Record your first script.
- View Results.



Practical Work

Outlines:

- JMeter prerequisites and installation.
- Main JMeter components.
- Record your first script.
- View Results.



JMeter prerequisites and installation.

- Here are the steps you should take in order to install JMeter:
 - Install the latest 64-bit JRE or JDK. This is important because JMeter is a pure Java application.
 - Go to Apache JMeter and find the Binary to download to your computer
 - Once downloaded, move this file to your preferred location, extract it and go to the folder, and then, the bin directory.
 - Take a look. You should see a series of scripts which can run JMeter in various modes.
- **Congratulations! You have all you need to start working on your test plan.**
- Detailed steps with screenshots are available [here](#).



Practical Work

Outlines:

- JMeter prerequisites and installation.
- Main JMeter components.
- Record your first script.
- View Results.



Main JMeter Components

JMETER

Thread Group

Samplers

**Logic
Controllers**

Listeners

**Configuration
Elements**

Assertions

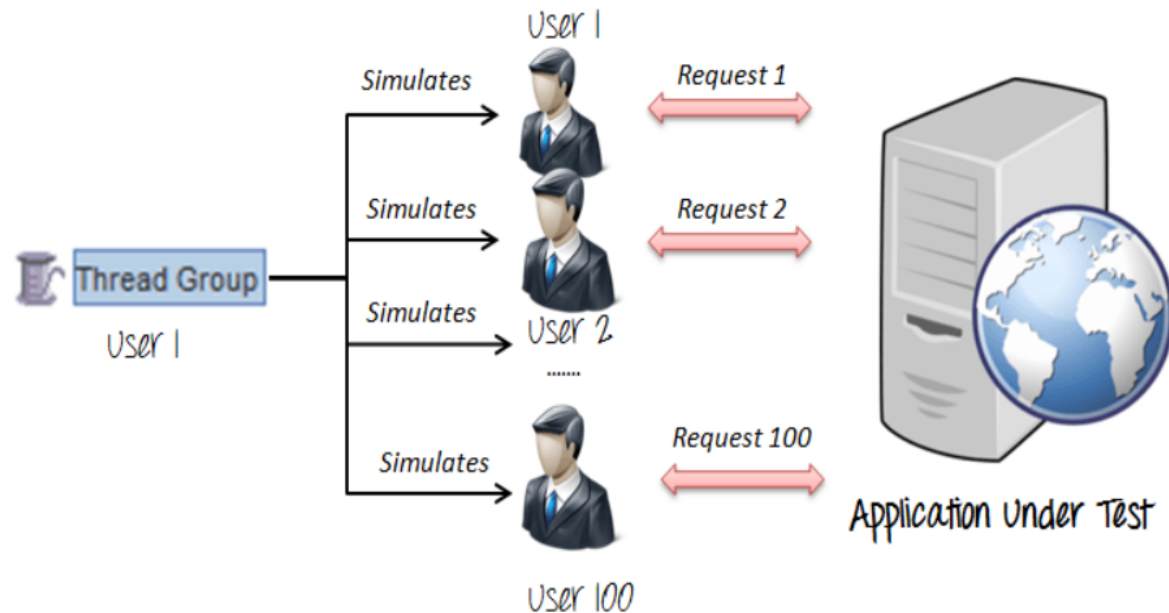
Timers

Processor



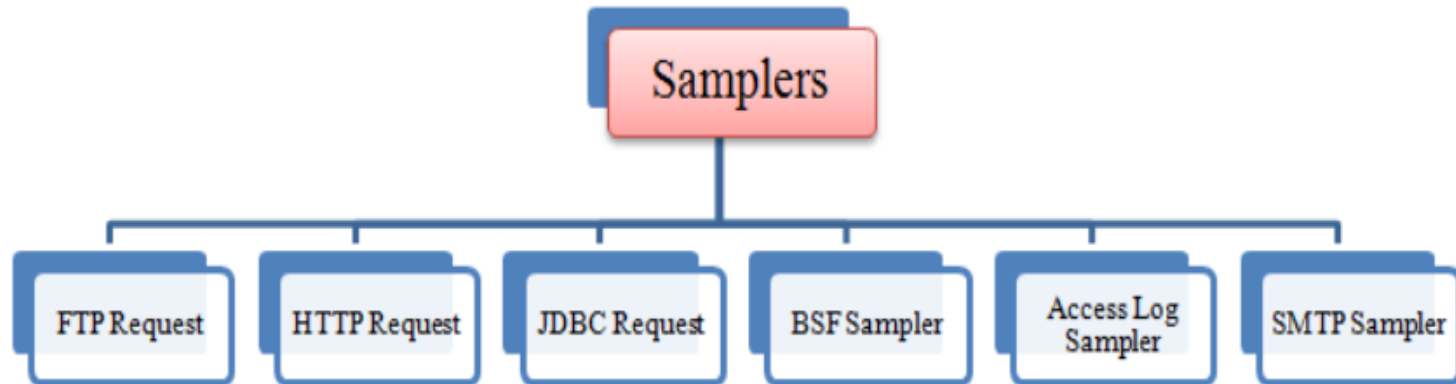
Main JMeter Components – Thread Group

- Thread Groups is a collection of Threads. Each thread represents one user using the application under test. Basically, each Thread simulates one real user request to the server.
- The controls for a thread group allow you to Set the number of threads for each group.
- For example, if you set the number of threads as 100; JMeter will create and simulate 100 user requests to the server under test



Main JMeter Components - Samplers

- JMeter supports testing HTTP, FTP, JDBC and many other protocols.
- We already know that Thread Groups simulate user request to the server, But how does a Thread Group know which type of requests (HTTP, FTP etc.) it needs to make?
The answer is Samplers
- The user request could be FTP Request, HTTP Request, JDBC Request...Etc.



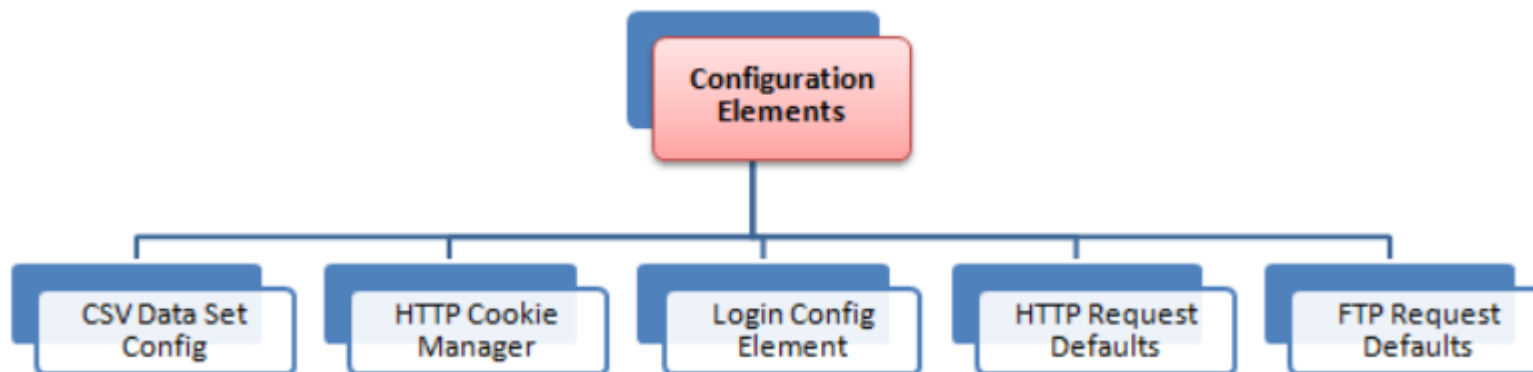
Main JMeter Components - Listener

- Listeners: shows the results of the test execution. They can show results in a different format such as a tree, table, graph or log file



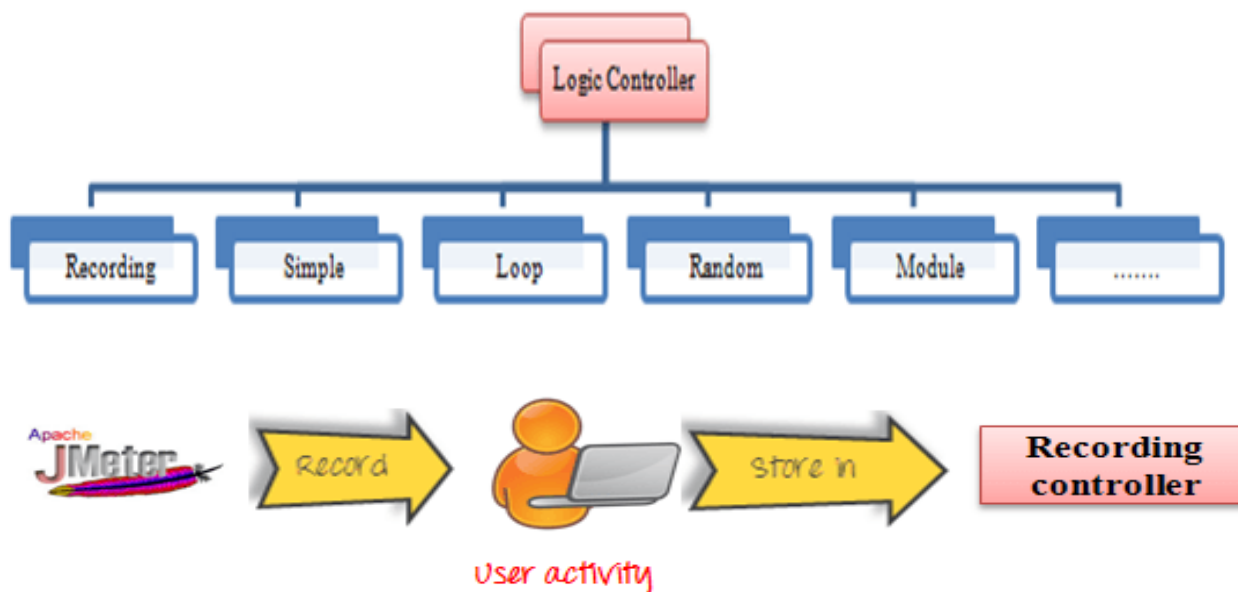
Main JMeter Components – Configuration elements

- Configuration elements set up defaults and variables for later use by samplers.



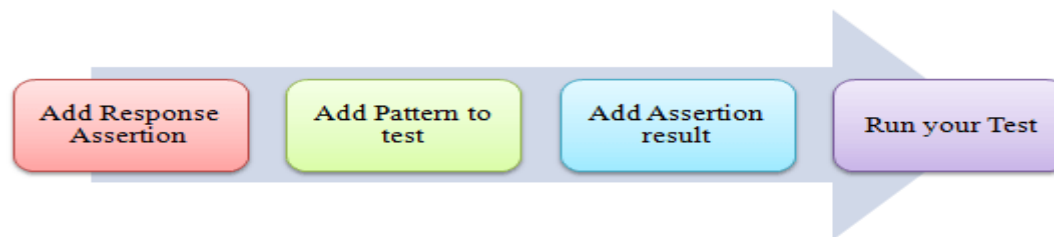
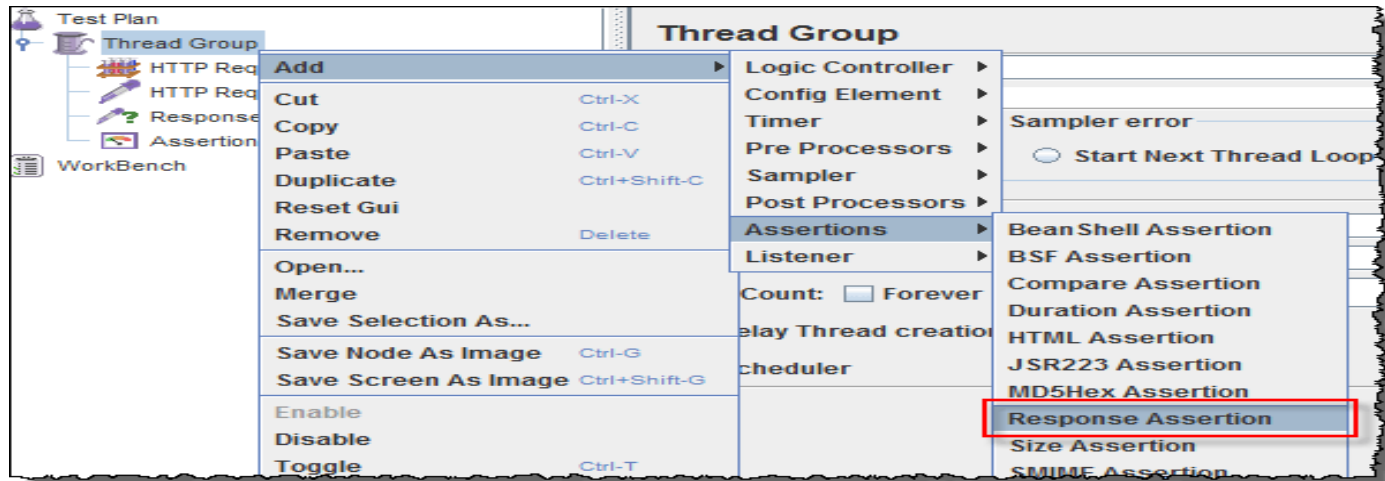
Main JMeter Components – Logic controller

- Logic Controllers let you define the order of processing request in a Thread. It lets you control “when” to send a user request to a web server. For example, you can use Random Controllers to send HTTP requests to the server randomly
- Logic Controllers determine the **order** in which user request is executed.
- Some commonly used Logic controllers are below:



Main JMeter Components – Assertions

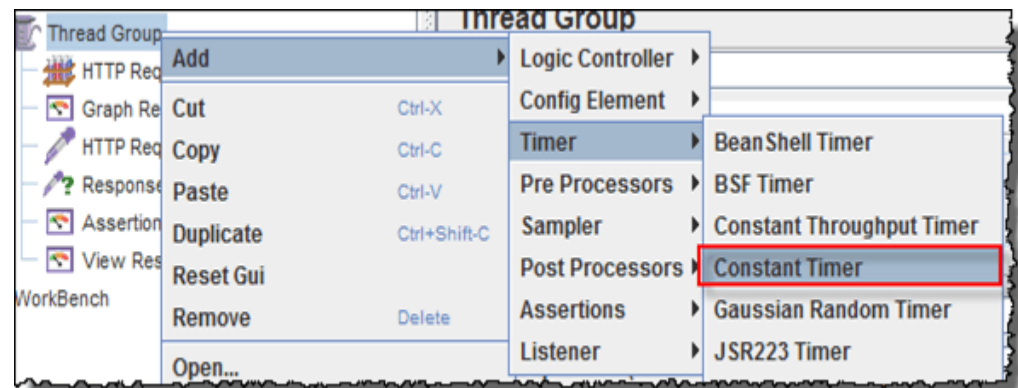
- Assertion help verifies that your server under test returns the **expected** results.



Main JMeter Components – Timers

- Timers allow JMeter to delay between each request which a thread makes.
- Also, in real life visitors do not arrive at a website all at the same time, but at different time intervals. So Timer will help mimic the real-time behavior.
- JMeter Timers:

- **Constant Timer**
- Gaussian Random Timer
- Uniform Random Timer
- BeanShell Timer
- BSF Timer
- JSR223 Timer
- How to Use Constant Timer



Practical Work

Outlines:

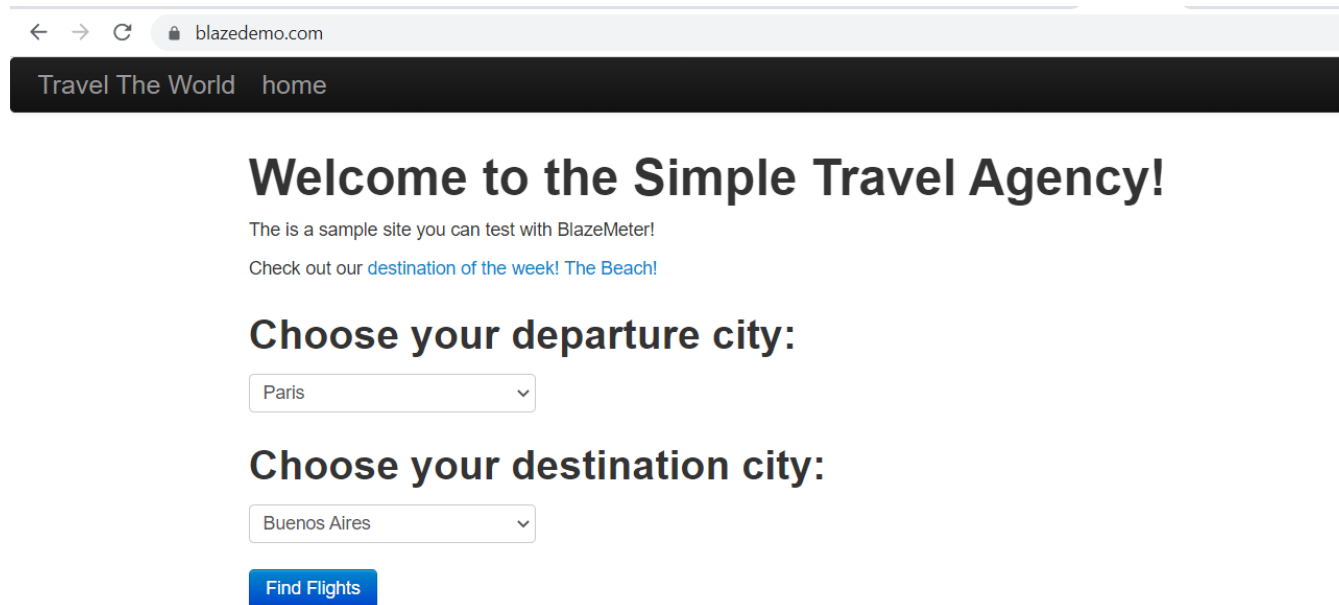
- JMeter prerequisites and installation.
- Main JMeter components.
- Record your first script.
- View Results.



Record your first Script

We will use below website to record our first JMeter Script:

<https://blazedemo.com/>



The screenshot shows a web browser window with the address bar displaying "blazedemo.com". The website has a dark header with the text "Travel The World" and a "home" link. The main content area features a large heading "Welcome to the Simple Travel Agency!" followed by a subtext "The is a sample site you can test with BlazeMeter!". Below this, there is a link "Check out our destination of the week! The Beach!". The form section is titled "Choose your departure city:" and includes a dropdown menu with "Paris" selected. Below that, it says "Choose your destination city:" with a dropdown menu showing "Buenos Aires". At the bottom of the form is a blue button labeled "Find Flights".



Record your first Script

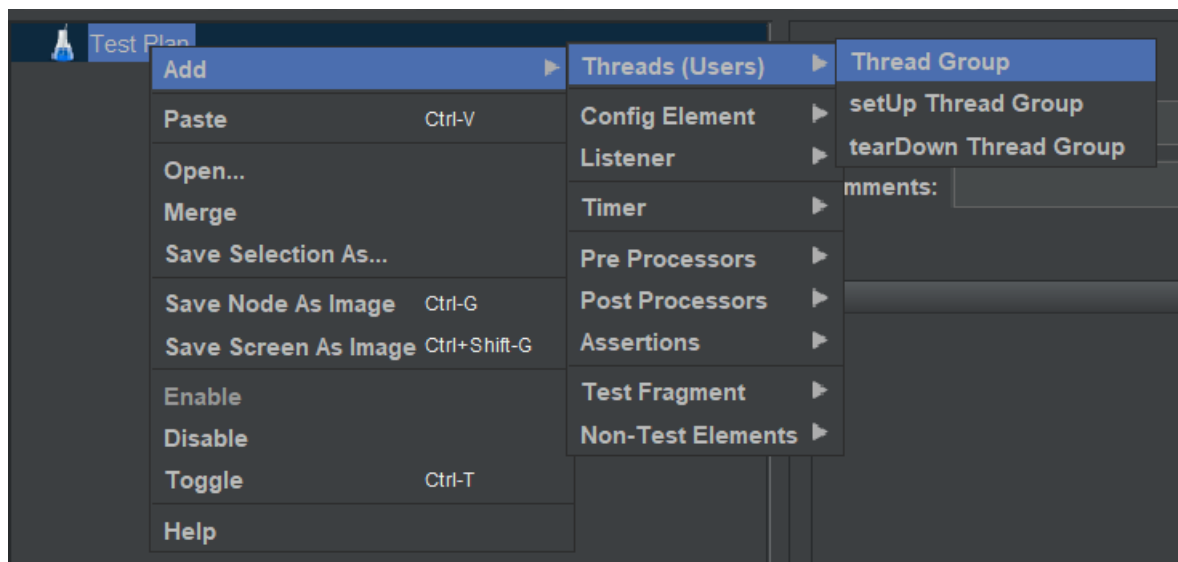
Step#1:

Run JMeter by running (jmeter.bat) that can be found under bin folder

 jmeter.bat	2/1/1980 12:00 AM	Windows Batch File	9 KB
--	-------------------	--------------------	------

Step#2:

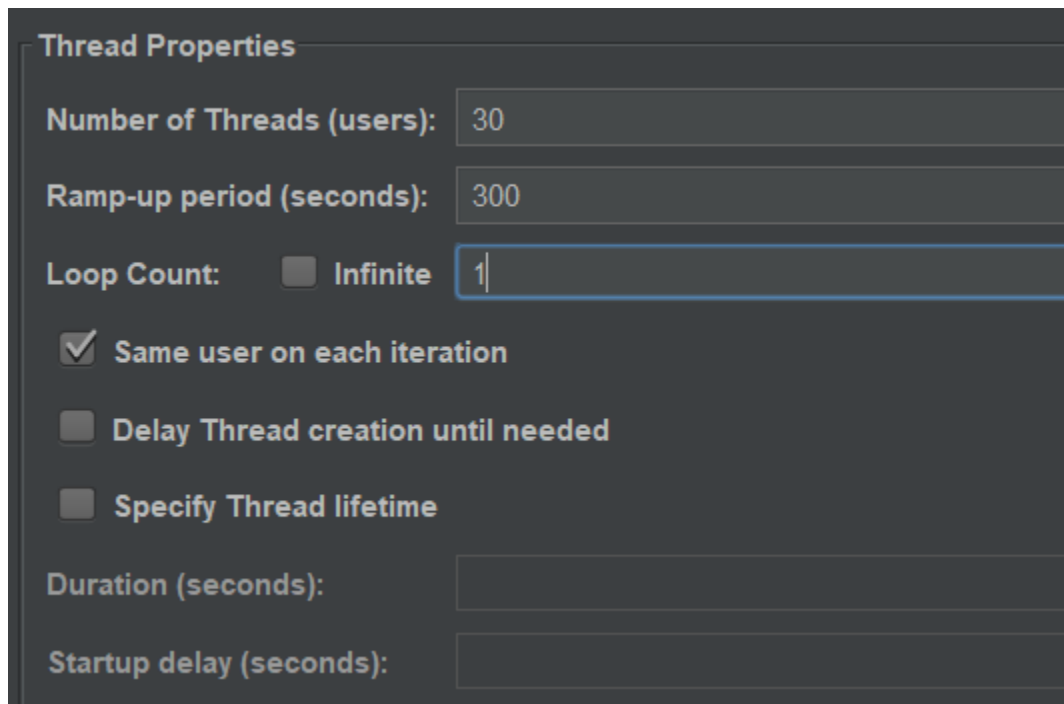
Add thread group as in below screenshot



Record your first Script

Step#3:

Configure Thread group prosperities as below



The screenshot shows the 'Thread Properties' dialog box in Apache JMeter. It contains the following fields and options:

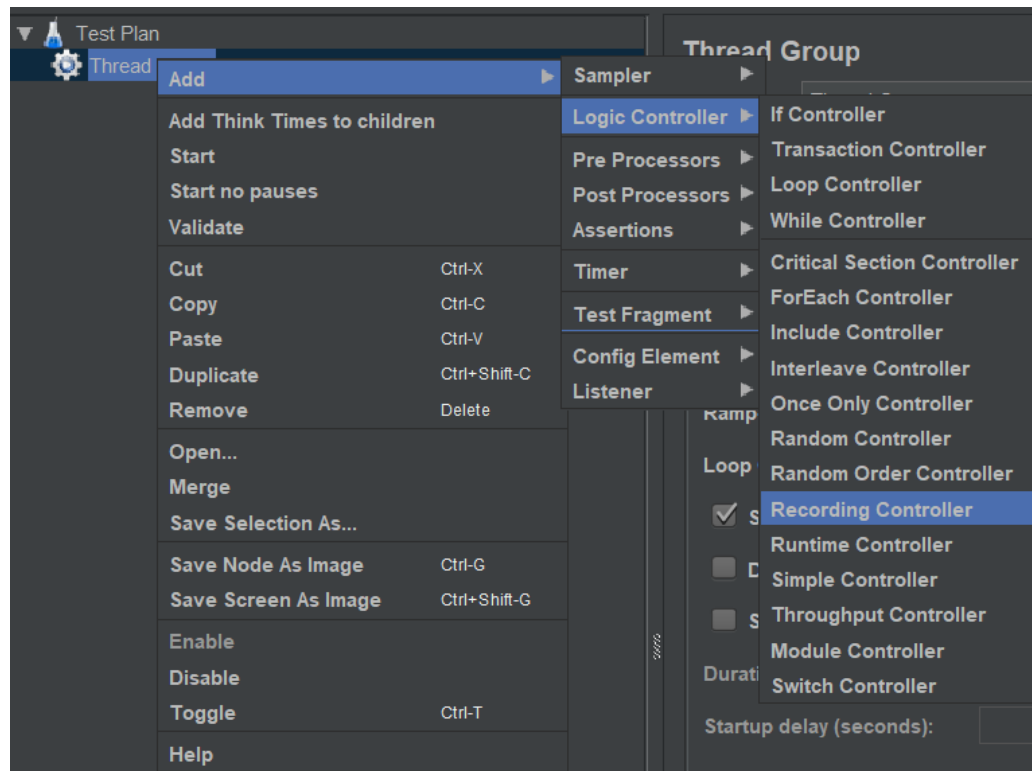
- Number of Threads (users):** 30
- Ramp-up period (seconds):** 300
- Loop Count:** ☐ Infinite, (The text input field is highlighted with a blue border.)
- ☒ Same user on each iteration
- ☐ Delay Thread creation until needed
- ☐ Specify Thread lifetime
- Duration (seconds):** (Empty text field)
- Startup delay (seconds):** (Empty text field)



Record your first Script

Step#4:

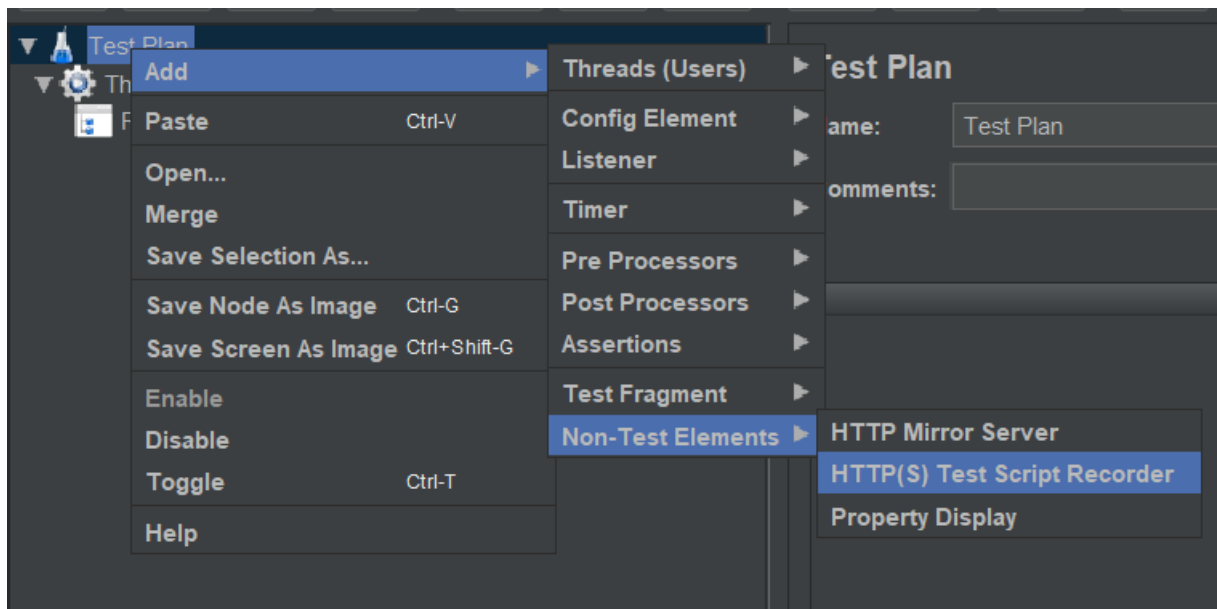
Add recording controller as in below



Record your first script

Step#5:

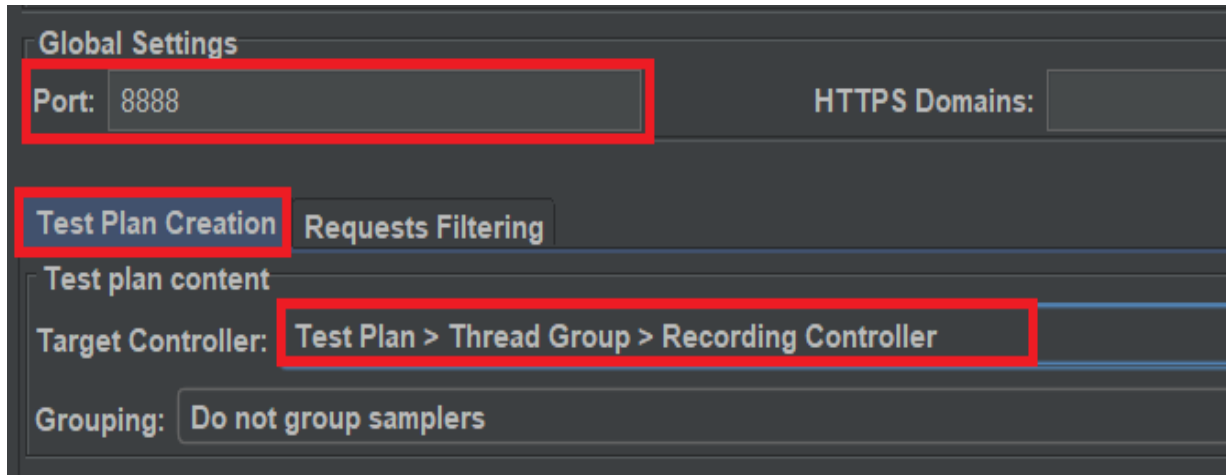
Add Http(s) test script recorder as in below:



Record your first script

Step#6:

Configure Http(s) test script recorder as below:



Global Settings

Port: 8888 HTTPS Domains:

Test Plan Creation Requests Filtering

Test plan content

Target Controller: Test Plan > Thread Group > Recording Controller

Grouping: Do not group samplers



Record your first script

Step#7:

Add below URL patterns to exclude in requests filtering

.*\js.*
.*\txt.*
.*\css.*
.*\png.*
.*\gif.*
.*\bmp.*
.*\ttf.*
.*\swf.*

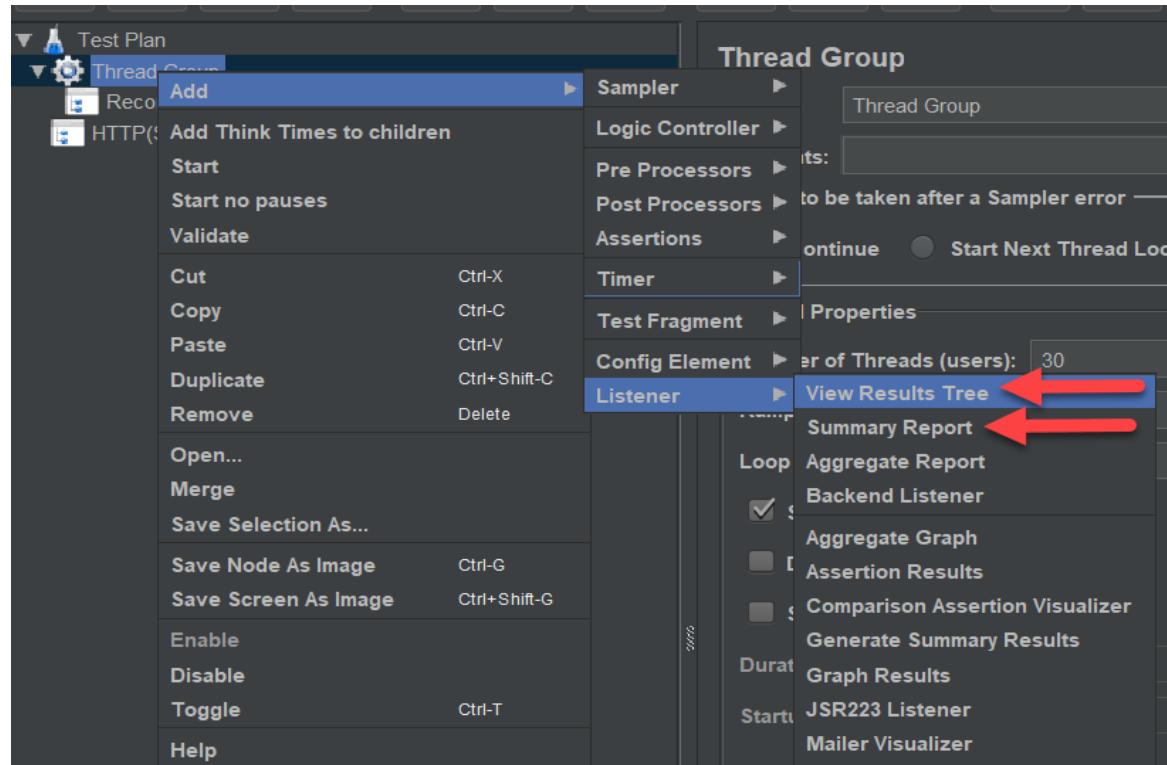
The screenshot shows the 'Test Plan Creation' window with the 'Requests Filtering' tab selected. It features a 'Content-type filter' section with 'Include' and 'Exclude' input fields. Below this are two sections: 'URL Patterns to Include' and 'URL Patterns to Exclude'. The 'URL Patterns to Exclude' section is highlighted with a red box and contains the following patterns: .*\.js.*, .*\.css.*, .*\.txt.*, .*\.png.*, and .*\.gif.*. At the bottom of this section, the 'Add' button is also highlighted with a red box. Other buttons visible include 'Delete', 'Add from Clipboard', and 'Add suggested Excludes'.



Record your first script

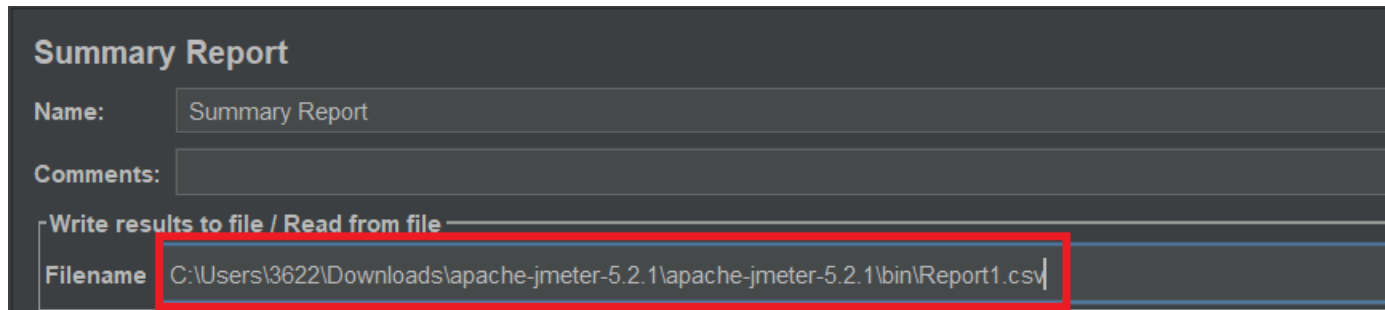
Step#8:

Add listeners as in below (view results Tree and Summary report)



Record your first script

Inside summary report we will add the URL where we want results to be saved which is the path where JMeter is located followed by csv file name as in below:



Summary Report

Name: Summary Report

Comments:

☒ Write results to file / Read from file

Filename: C:\Users\3622\Downloads\apache-jmeter-5.2.1\apache-jmeter-5.2.1\bin\Report1.csv



Record your first script

Step#9:

From Firefox network settings, configure connection settings as in below screenshots:

Connection Settings

Configure Proxy Access to the Internet

- ☐ No proxy
- ☐ Auto-detect proxy settings for this network
- ☐ Use system proxy settings
- ☒ Manual proxy configuration

HTTP Proxy

localhost

Port

8888

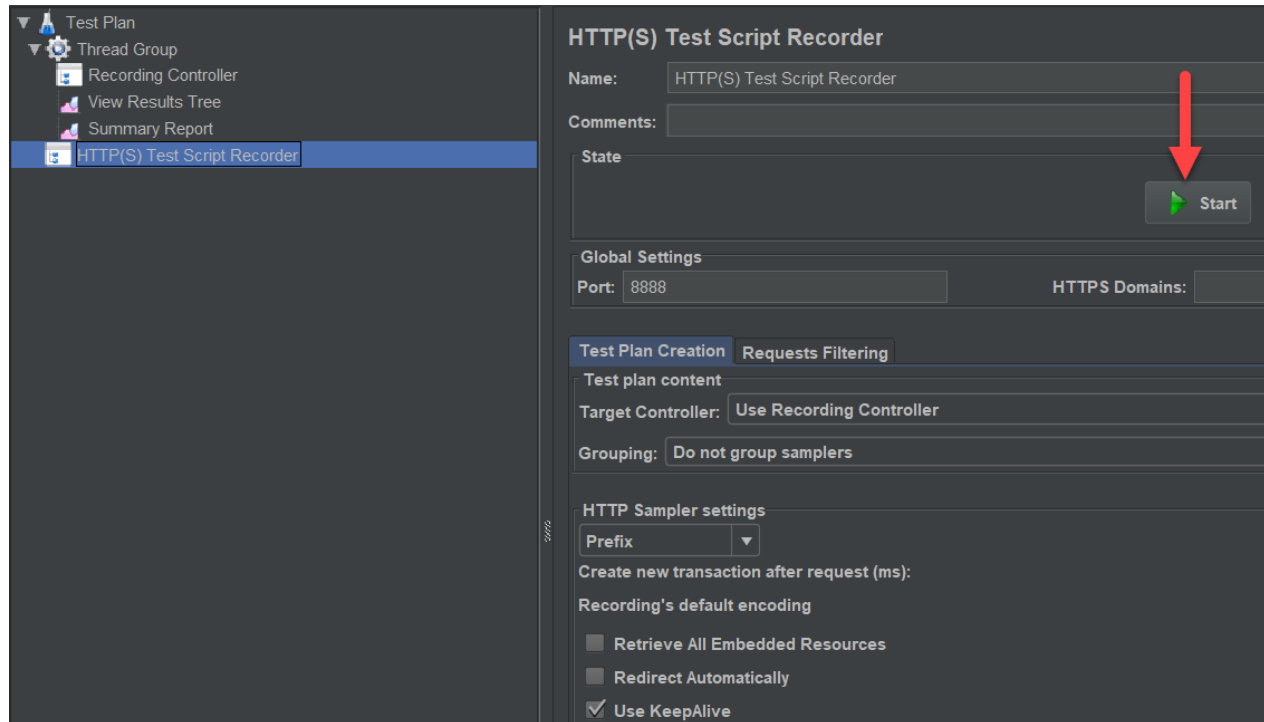
☒ Also use this proxy for HTTPS



Record your first script

Step#10:

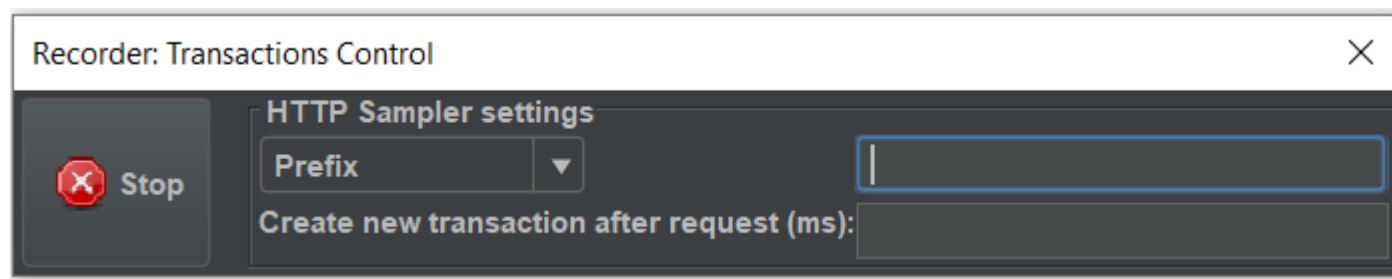
Start your recording by clicking on start from Http(s) test script recorder:



Record your first script

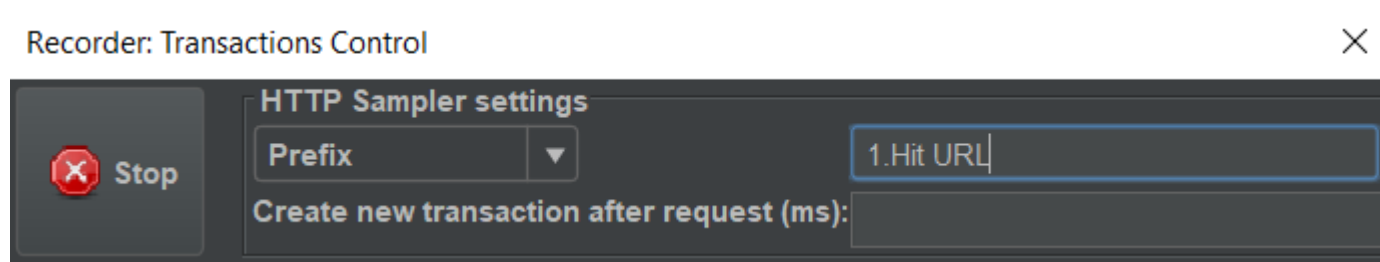
Step#11:

Below dialog will appear, we will use it to label each step in our recording and to stop the recording when we finish



Step#12:

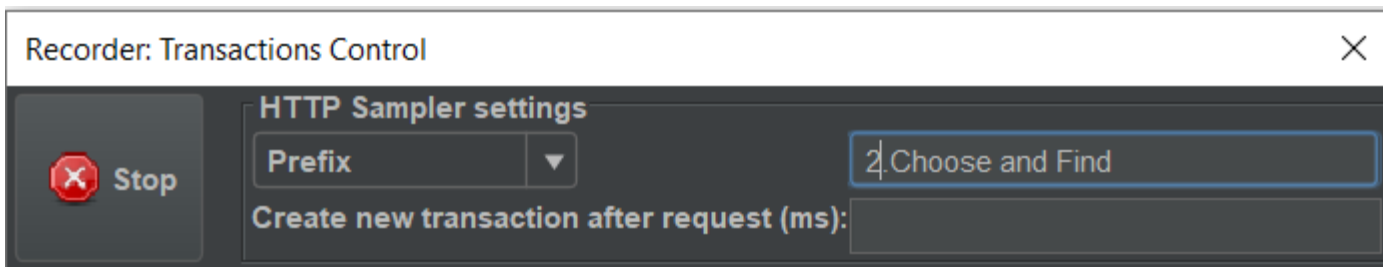
Open the Firefox browser, and type inside the prefix 1.Hit URL, then enter the URL <https://blazedemo.com/> in Firefox browser and press enter



Record your first script

Step#13:

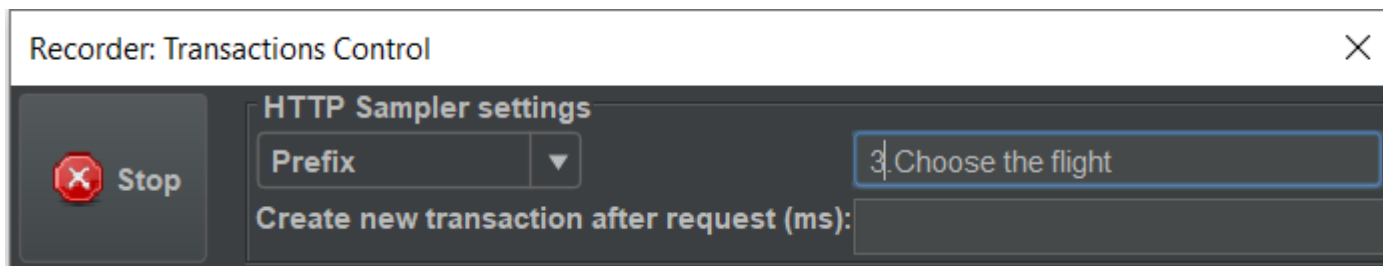
Type inside the prefix 2.Choose and Find then Choose departure city and destination city then press on find flights



The screenshot shows the 'Recorder: Transactions Control' window. On the left is a 'Stop' button with a red 'X' icon. The main area is titled 'HTTP Sampler settings'. It contains a 'Prefix' dropdown menu, a text input field containing '2.Choose and Find', and a 'Create new transaction after request (ms):' field.

Step#14:

Type inside the prefix 3.Choose the flight then Choose one of the flights



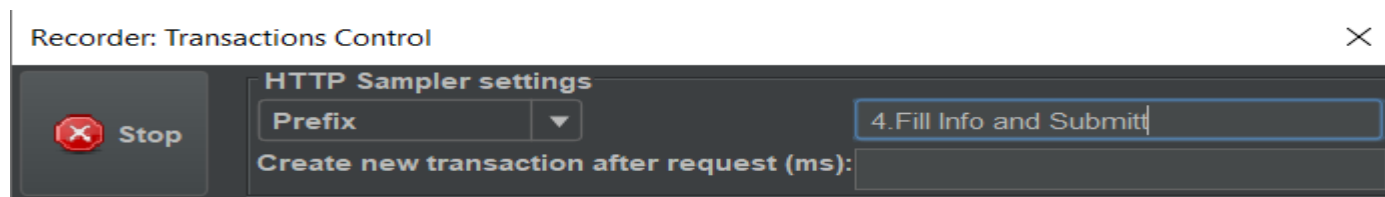
The screenshot shows the 'Recorder: Transactions Control' window. On the left is a 'Stop' button with a red 'X' icon. The main area is titled 'HTTP Sampler settings'. It contains a 'Prefix' dropdown menu, a text input field containing '3.Choose the flight', and a 'Create new transaction after request (ms):' field.



Record your first script

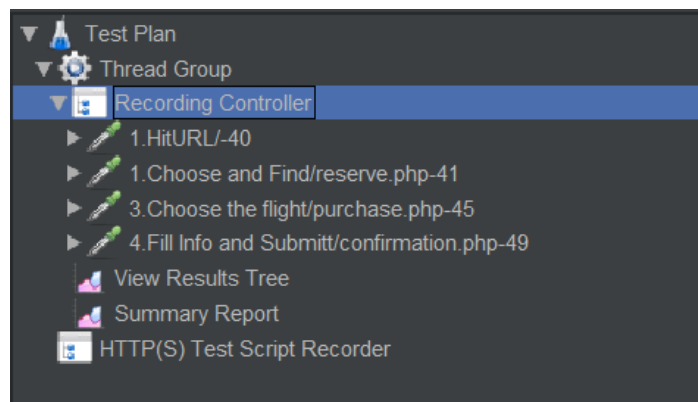
Step#15:

Type inside the prefix 4.Fill Info and Submit then fill all the needed information and press Purchase flight



Step#16:

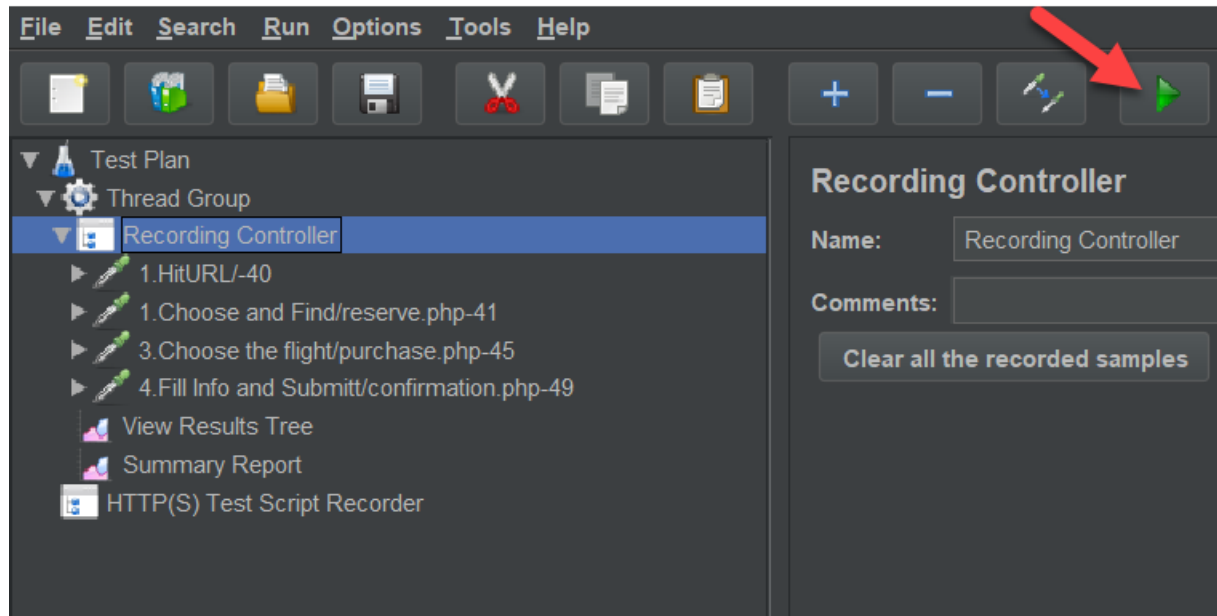
Click on the stop button inside the recorder and will find your first recorded script under Recording controller:



Record your first script

Step#17:

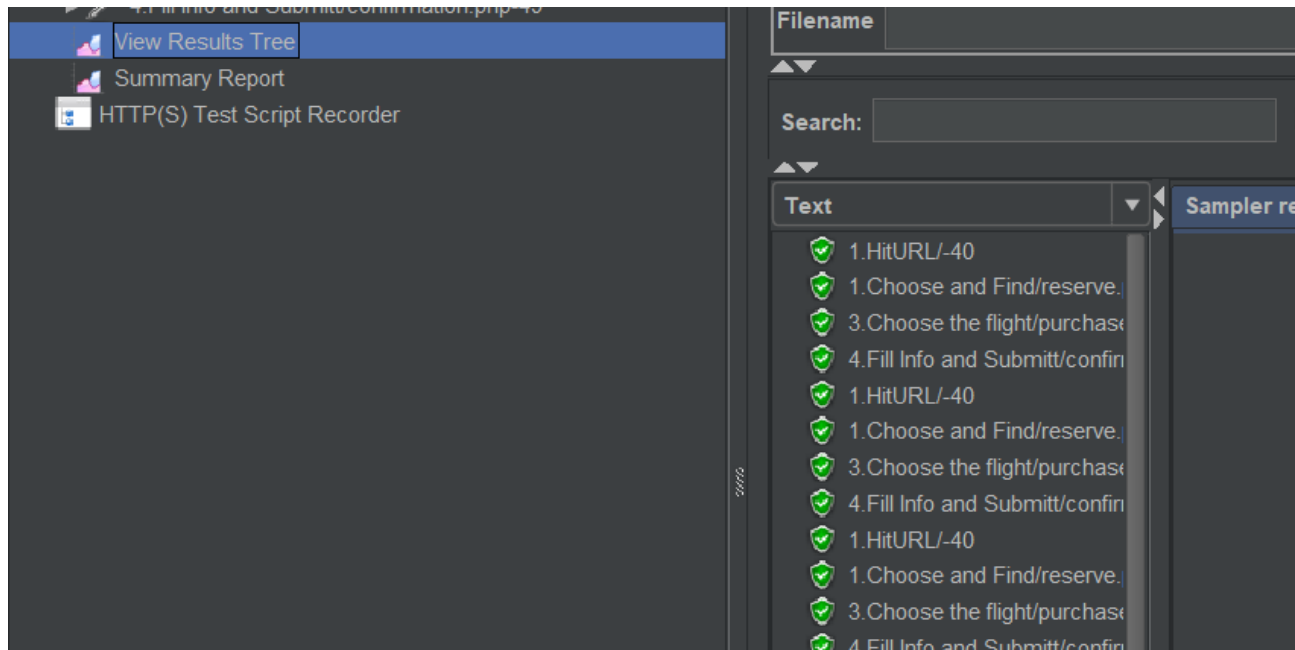
Press on the run button to run your script with the same thread group configurations we already did



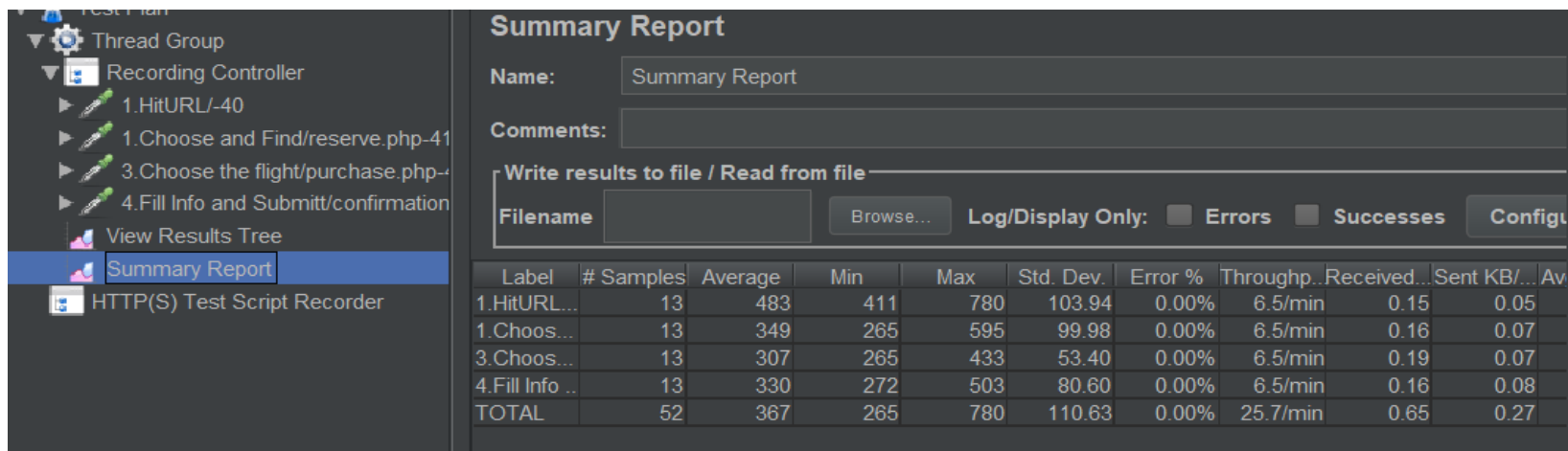
Record your first script

Step#18:

We can view results during and after the run from the listeners we already added as in below:



Record your first script



The screenshot shows the JMeter Summary Report interface. On the left is a tree view with 'Thread Group' expanded, showing a 'Recording Controller' with four steps: '1.HitURL/-40', '1.Choose and Find/reserve.php-41', '3.Choose the flight/purchase.php-', and '4.Fill Info and Submitt/confirmation'. Below this are 'View Results Tree', 'Summary Report' (selected), and 'HTTP(S) Test Script Recorder'. The main panel displays the 'Summary Report' for a test named 'Summary Report'. It includes fields for 'Name' and 'Comments', and a section for 'Write results to file / Read from file' with a 'Filename' field and a 'Browse...' button. Below this is a table with performance metrics.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughp..	Received...	Sent KB/...	Av
1.HitURL...	13	483	411	780	103.94	0.00%	6.5/min	0.15	0.05	
1.Choos...	13	349	265	595	99.98	0.00%	6.5/min	0.16	0.07	
3.Choos...	13	307	265	433	53.40	0.00%	6.5/min	0.19	0.07	
4.Fill Info ..	13	330	272	503	80.60	0.00%	6.5/min	0.16	0.08	
TOTAL	52	367	265	780	110.63	0.00%	25.7/min	0.65	0.27	

Congratulations !
We are done with recording our first script and running it !



Practical Work

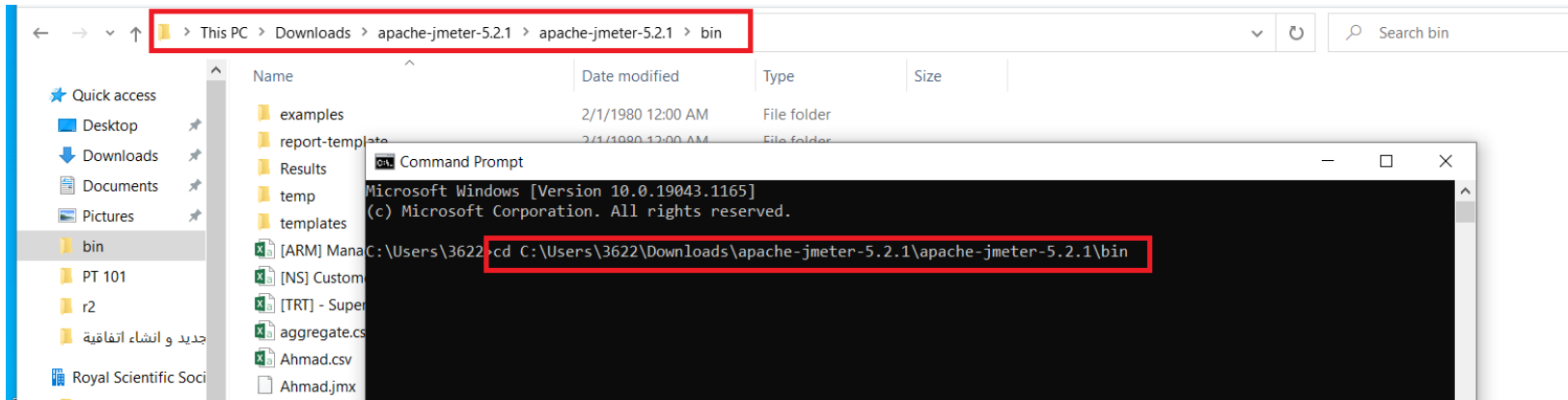
Outlines:

- JMeter prerequisites and installation.
- Main JMeter components.
- Record your first script.
- View Results.



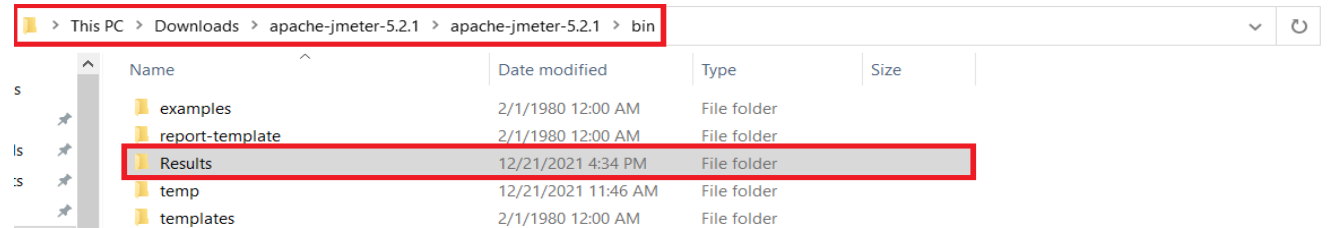
View Results

- The dashboard generator is a modular extension of JMeter. Its default behavior is to read and process samples from CSV files to generate HTML files containing graph views. It can generate the report at end of a load test or on demand.
- To generate reports for our script we will follow below steps:
 - **Step#1:** Open CMD terminal.
 - **Step#2:** Change directory to the path where csv results file saved using following command: **cd path**

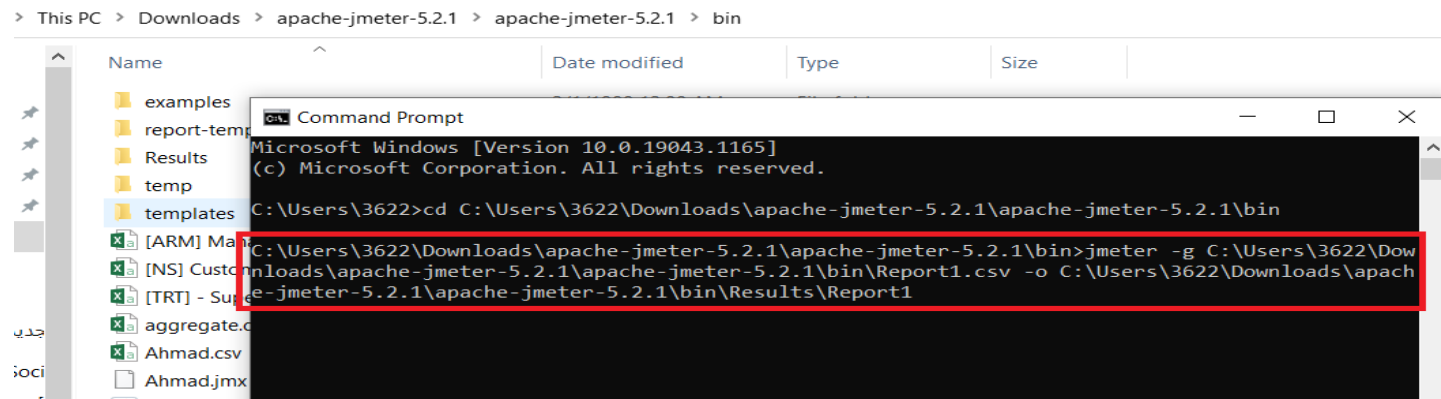


View Results (each step in a slide with screenshot)

➤ **Step#3:** Create Results folder inside bin folder.

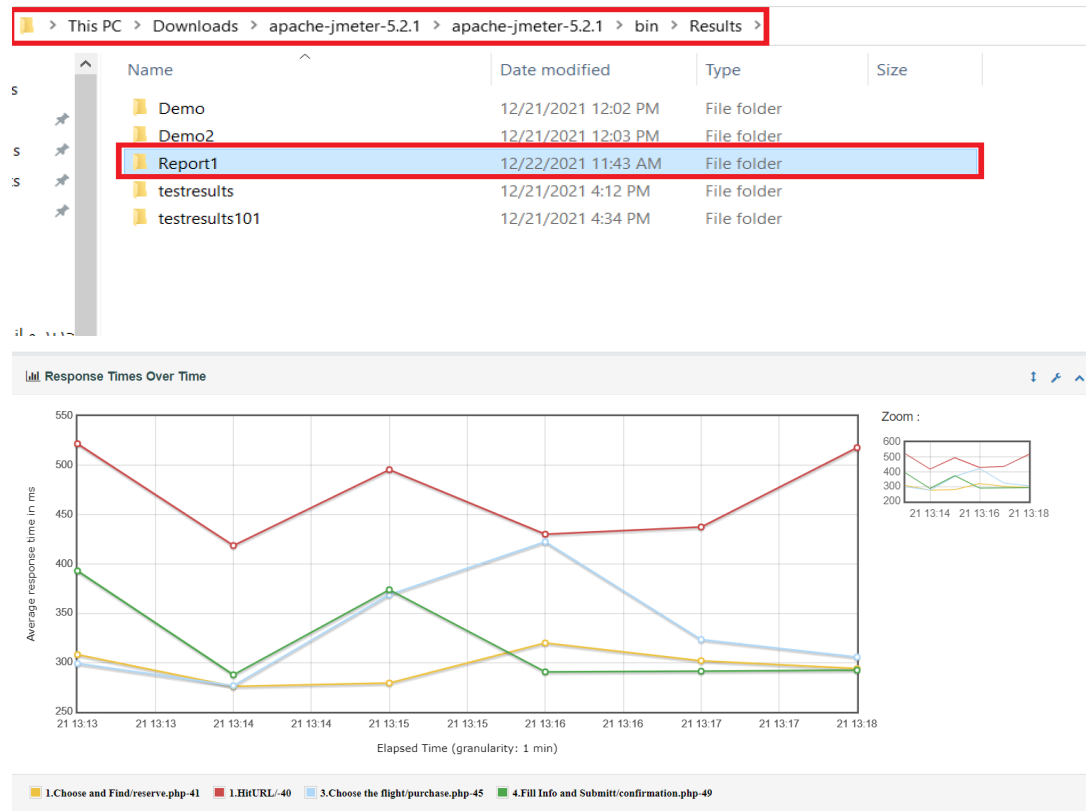


➤ **Step#4:** Use the following command to generate reports: `jmeter -g path\FileName.csv -o path\Results\FileName`



View Results (each step in a slide with screenshot)

- **Step#5:** Open the specified path and check index.html file inside the created folder under results folder.



The Final Word..

- Whether it is a mobile app or a desktop application, employing **performance testing services** is a must. From scalability to speed, reliability, and robustness.
- In Jordan, there is a **great demand for Performance testers** with a very good salary ranges.
- If you found this topic interesting, and you want to be an **officially performance tester**:





الْجَمْعِيَّةُ الْعِلْمِيَّةُ الْمَلَكِيَّةُ
Royal Scientific Society

Thank you

For further information please contact NSQAC on:

Tel: +962-6-5344701, ext.: 2552, 2453, 2556

Mobile: +962-78-8019392

Email: nsqac.rss@rss.jo

PO Box 1438, Amman 11941, Jordan

T (+962) 6 534 4701 | F (+962) 6 534 4806 | www.rss.jo