

# Performance Testing

Testing Mavens  
Software Testing Redefined



# Functional vs Non-Functional Requirements

FUNCTIONAL	NONFUNCTIONAL
WHAT a system does	HOW <i>well</i> the system does it within design & resource constraints

Non-Functional Requirement – A software requirement that describes **not what** the software will do, **but how** the software will do it.

# Functional vs Non-Functional Requirements ...

What is functional testing?	What is nonfunctional testing
When input credentials are valid, user can login	After login, the dashboard loads within 3 seconds
When email notifications are on, and user receives a new message, an email notification is sent	The email notification is sent within 5 minutes
When a JPG file under 1MB is uploaded, the uploader accepts the file	When eight files or less (each under 1MB) are uploaded at the same time, the queues all
When the settings menu item is clicked, the settings page loads	The settings page has a matching appearance to the rest of the GUI

# What is Performance Testing ?

Performance Testing is a systematic testing approach to validate the ***performance*** of an application under ***load***.

## ***Performance***

The word 'Performance' refers to the behavior of an application in terms of scalability, stability, and speed.

## ***Load***

'Load' refers to the burden on an application in terms of the real-world user count.

*For example, when a user sends a message to another user, the message should be successfully delivered to the recipient. This is a functional check. Additionally, the message should be delivered within 5 seconds of sending. This is a performance check.*

# Why Performance Testing ?

Imagine an e-commerce website that expects a surge in traffic during a holiday season sale. Performance testing would involve simulating large number of concurrent users accessing the website, adding items to their carts, and completing the checkout process. The purpose is to evaluate if the website can handle the anticipated load without experiencing slow response times, errors, or crashes.

1. To Identify whether the application can handle the peak load
2. Observe the behavior of the application in terms of response time
3. To check whether the resources (CPU, Memory, and Disk) do not breach the defined performance limit
4. To identify if there is any bottleneck

# Importance of Performance Testing

- **To measure the responsiveness of the website/system :** Performance testing verifies how quickly the system responds to user actions or requests, ensuring a satisfactory user experience.
- **Identify Bottlenecks :** It helps identify performance bottlenecks and weaknesses in the system, enabling optimization for better performance.
- **Plan for Scalability :** Performance testing helps determine the system's capacity to handle increased user demand or larger dataset aiding in capacity planning and scalability.
- **Enhance User Experience :** By resolving performance issues like slow response times or crashes, Performance testing improves user satisfaction and reduces frustration.
- **Ensure Reliability :** Performance testing validates the system's stability, minimizing downtime and ensuring uninterrupted availability.
- **Comply with regulations :** In some industries, Performance testing ensures compliance with performance-related standards or SLAs.

# What does Performance Testing measure ?

## **CPU & Memory utilization :**

- It is the percentage / measure of CPU & Memory capacity utilized in processing the requests.

## **Response times :**

- It is the total time between sending the request and receiving the response. Better the response time, better the performance of website / application.

## **Throughput :**

- It measures the number of transactions an application can handle in a second, or in other words, it is the rate at which a network or computer receives the requests per second.

## **Average latency / Wait time :**

- It is the time spent by a request in a queue before getting processed.

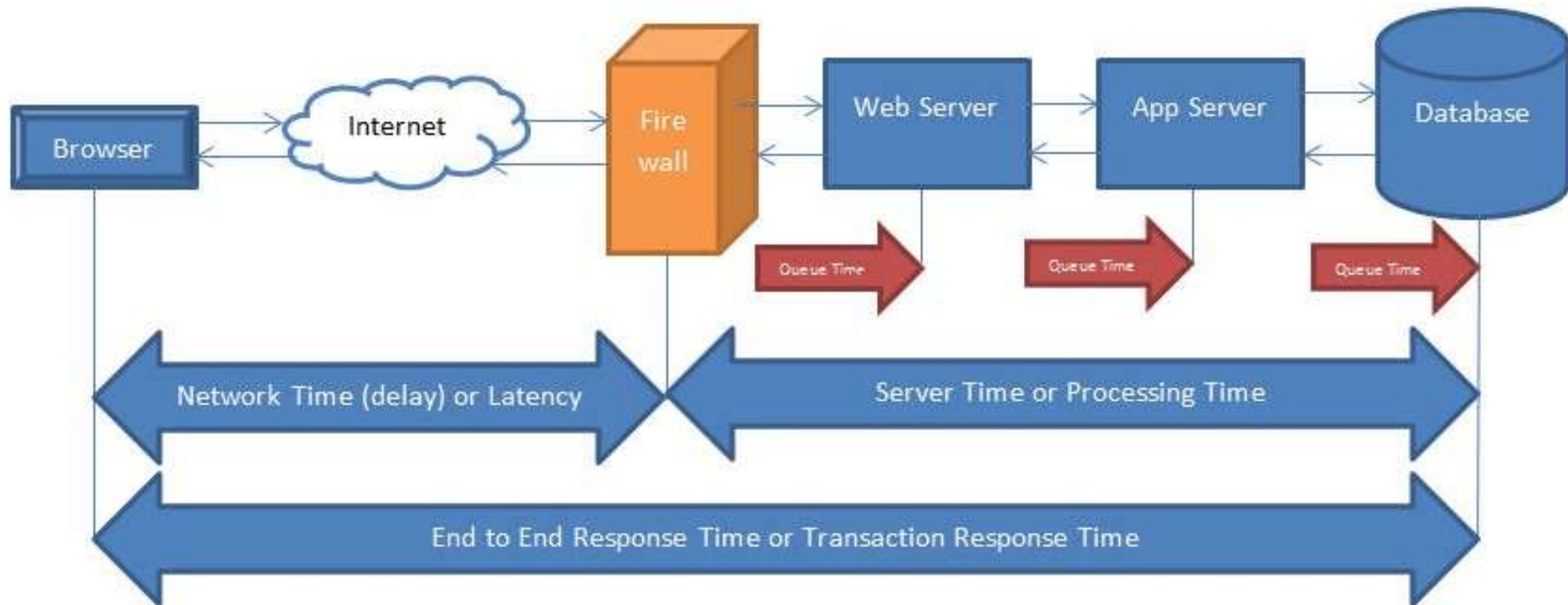
## **Bandwidth :**

- It is the measurement of the volume of data transferred per second.

## **Error rate :**

- It is the percentage of requests resulting in errors compared to the total number of requests.

## What does Performance Testing measure ?





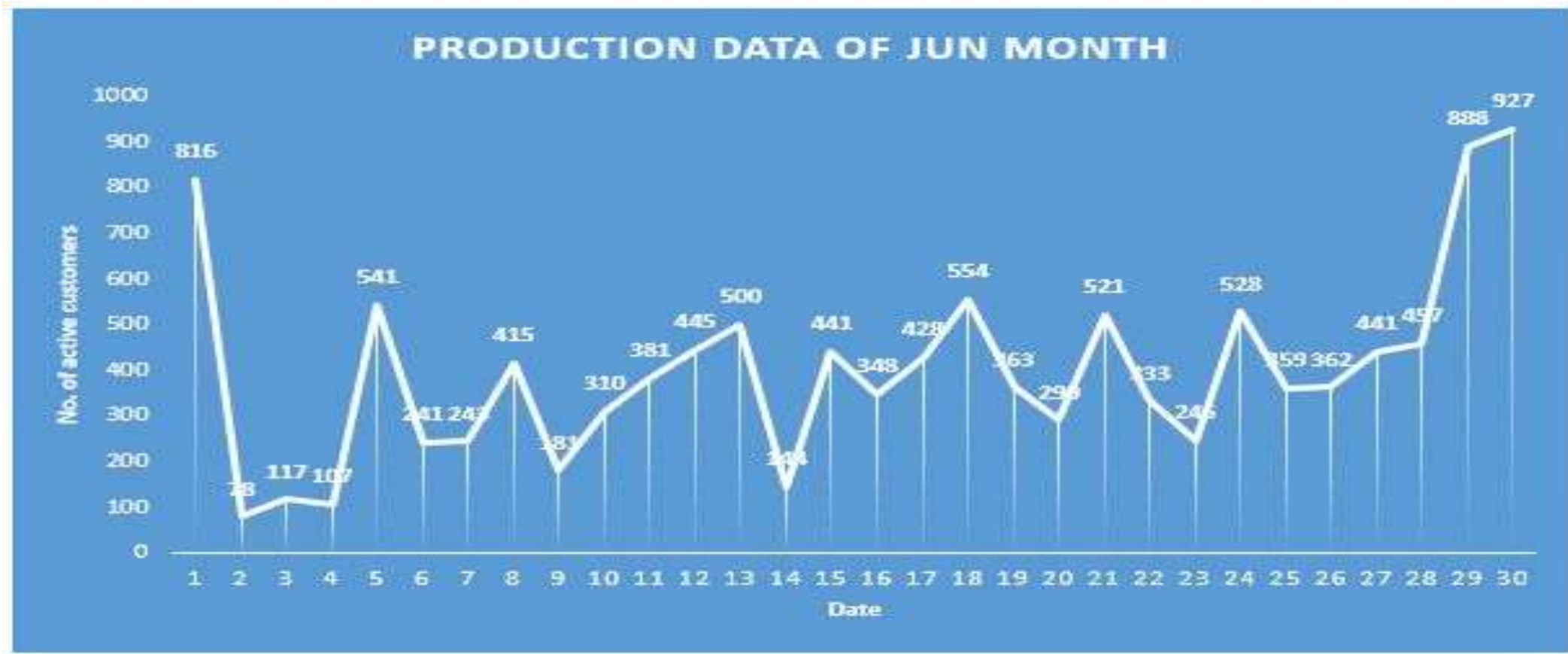
# Types of performance testing

Performance Testing is a very generic term. Each type of performance testing serves a specific purpose and provides valuable insights into different aspects of system performance. Depending on the project requirements, a combination of these testing types may be applied to thoroughly monitor and optimize the system's performance.

1. **Load Test** : A Load Test is a type of non-functional test that verifies the performance of an application or system under a peak user / volume condition. This test also validates the software system's resource usage, stability, and reliability under **peak load**.
2. **Stress Test** : A stress Test is a type of non-functional test that verifies the performance of an application or software system at a **futuristic load**. The futuristic load is a predicted load that should be handled by an application in the future with existing software and hardware. The defined load for a Stress test is always more than a Load Test.
3. **Soak Test** : A Soak test, also known as endurance testing or stability testing, or capacity testing, is a type of software testing that involves subjecting a system or application to a sustained load or stress over an extended period. Generally, the test duration of the soak test is between 8 to 24 hours, but it may vary as per project requirements. A longer period performance test with an **average load** provides information about the behavior of the Garbage collector and memory management.

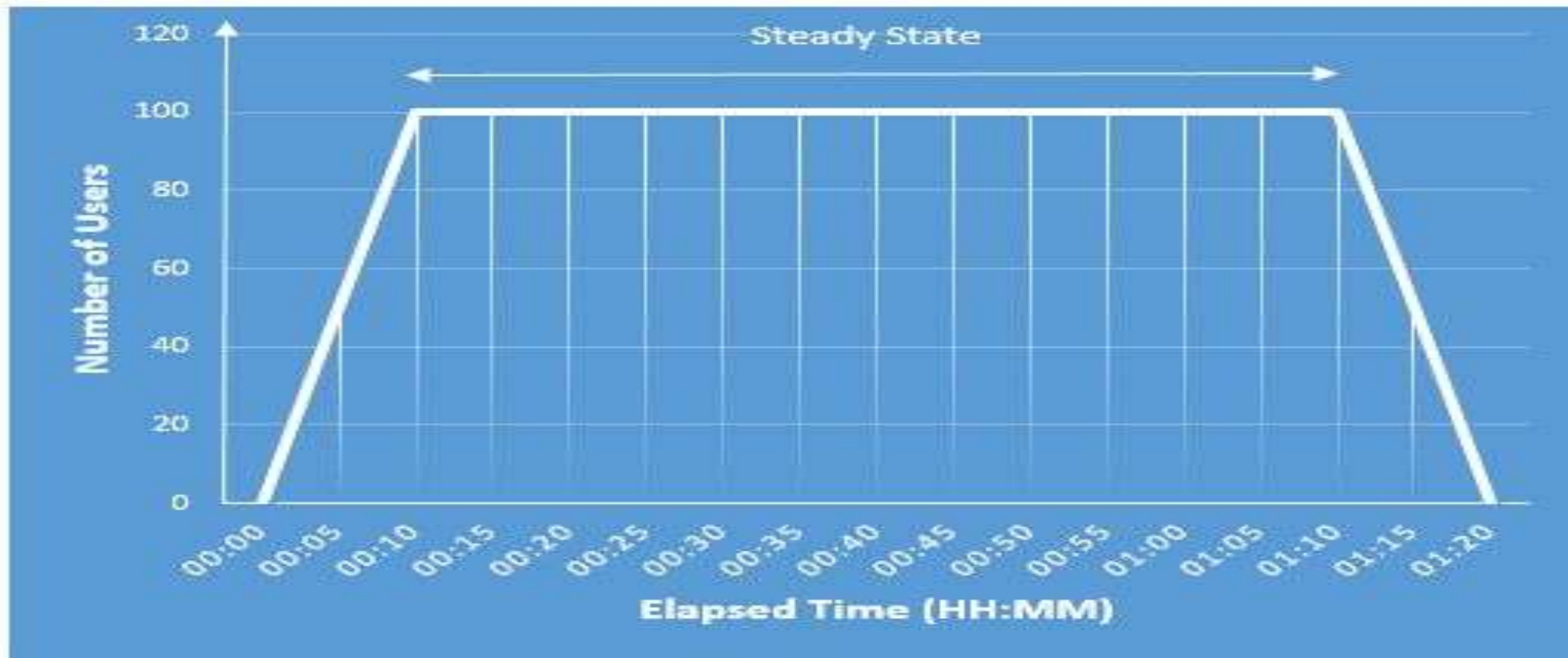
# Load Testing ...

What is Peak Load?



This graph shows the number of active customers per day in Jun. The highest number of active customers is 927 on 30th Jun. Hence 927 is the peak load.

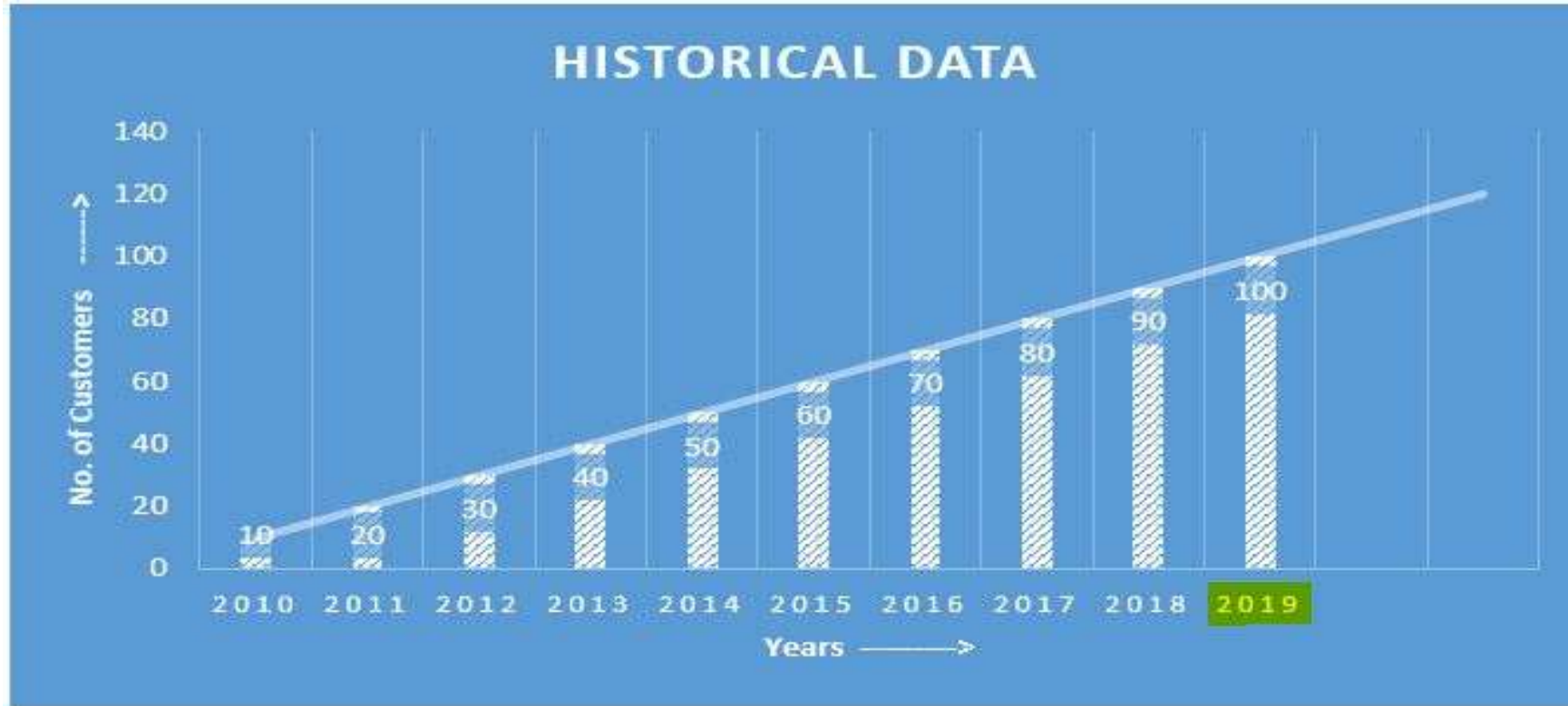
# Load Testing ...



The above graph has a steady state of 1 hour along with 10 minutes ramp-up and 10 minutes ramp-down period. Therefore, the test will run for 1 hour and 20 minutes.

# Stress Testing ...

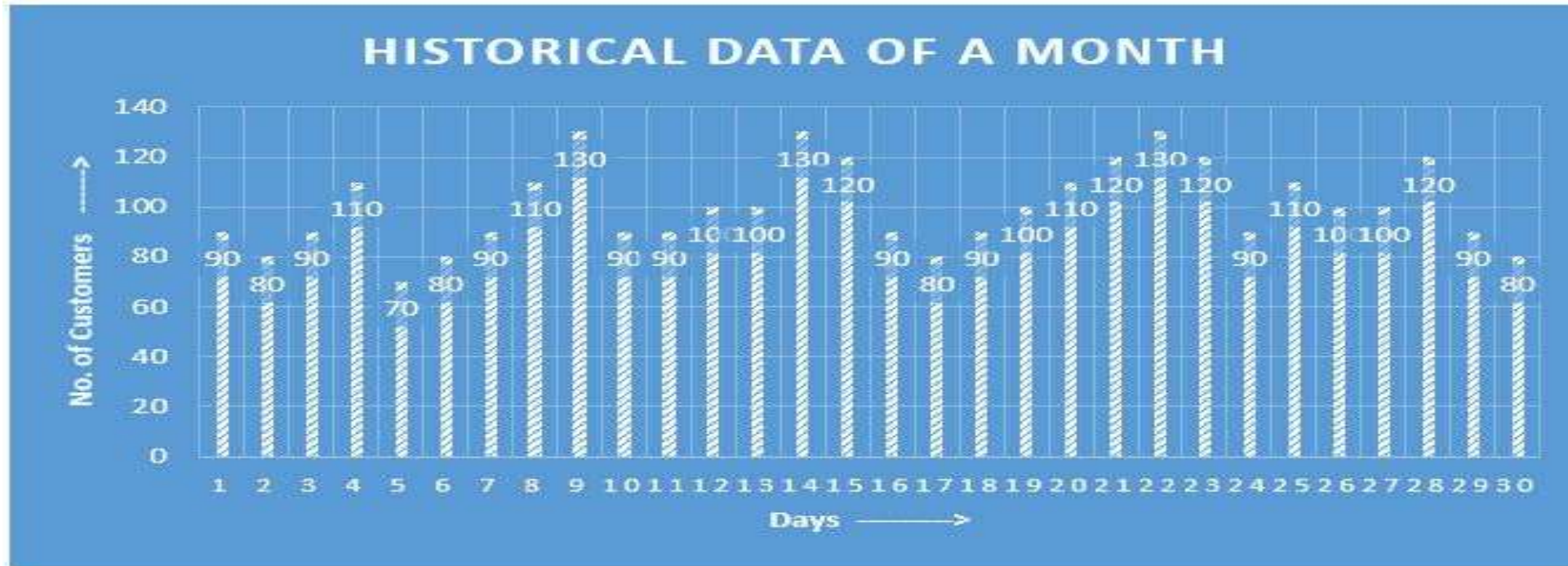
*How to calculate the Future Load ?*



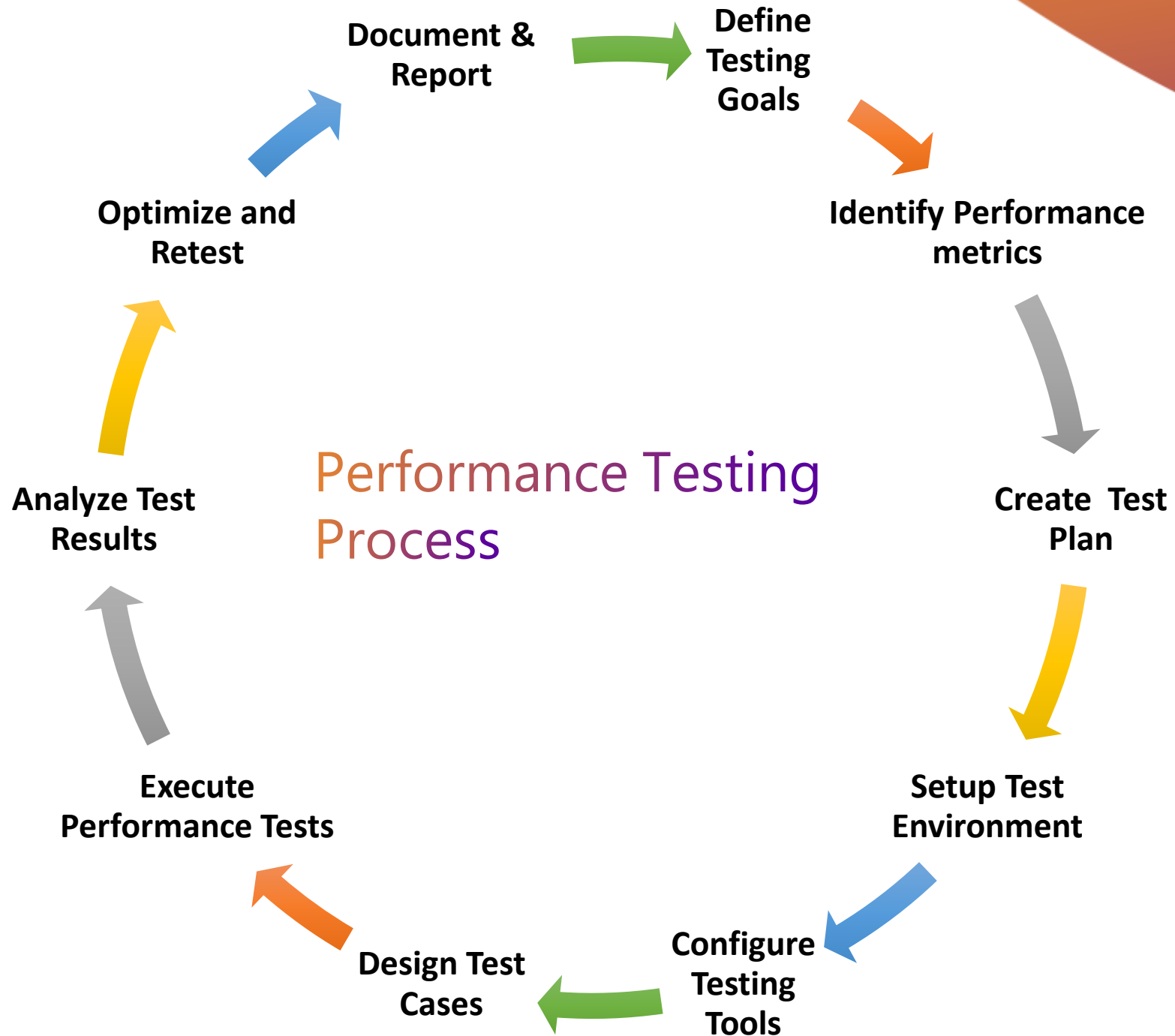
The future load is calculated with the help of historical data. The business analyst predicts the numbers by referring to previous years' statistics.

# Soak Testing ...

*How to calculate the Average Load ?*



The Average load is calculated with the help of historical data. The business analyst predicts the numbers by referring to the previous day's statistics.





# Performance Testing Process ...

- Define Testing Goals: Determine the specific objectives of the performance test. Identify what aspects of the system's performance you want to evaluate, such as response time, throughput, scalability, or resource utilization.
- Identify Performance Metrics: Determine the key performance metrics that will be measured during the test. For example, response time, transactions per second, error rates, or CPU/memory usage.
- Create Test Plan: Develop a detailed test plan that outlines the test scenarios, test environment, workload patterns, and performance goals. Define the test scenarios based on different user behaviours, transaction volumes, or system configurations.
- Set up Test Environment: Prepare the test environment that closely resembles the production environment in terms of hardware, software, and network configuration. Install and configure the necessary software, databases, and network infrastructure.
- Configure Performance testing tools: Select and configure the appropriate performance testing tools according to your requirements. Popular tools include Apache JMeter, LoadRunner, or Gatling. Set up the monitoring and profiling tools to collect performance metrics during the test execution.

# Performance Testing Process ...

- **Design Test Cases:** Create test cases that represent the anticipated user interactions with the system. Define the input data, user actions, and expected system responses. Test cases should cover a range of scenarios to simulate realistic usage patterns.
- **Execute Performance Test:** Run the performance tests according to the test plan. Start with a small workload and gradually increase the load to simulate real-world scenarios. Monitor and measure the system's performance using the defined metrics.
- **Analyze Test Results:** Collect and analyze the performance data obtained during the test execution. Identify any performance bottlenecks, scalability issues, or areas for improvement. Compare the test results against the predefined performance goals.
- **Optimize and Retest:** Based on the findings from the analysis, optimize the system, such as tuning configurations, optimizing code, or adding resources. Re-test the system to validate the effectiveness of the optimizations and ensure the performance goals are met.
- **Document and Report:** Document the entire performance testing process, including test plans, test cases, test results, and any findings. Prepare a comprehensive report summarizing the performance test outcomes, highlighting any issues, and providing recommendations for improvement.



Q & A



# Performance Testing Tools

- The primary goal of performance testing in general is to determine the **Speed**, **Stability** and **Scalability** of the application under test.
- Evaluating application performance with real users can lead to scalability limitations, inconsistent execution, inadequate metrics, inefficiency in time and cost, and inability to simulate realistic loads.
- Performance testing tools overcome these disadvantages.
- Performance testing tools can be open source or commercial.
- Few examples of popular Performance testing tools include :
  - Apache JMeter
  - HP LoadRunner
  - BlazeMeter
  - LoadNinja
  - Gatling
  - k6
  - Locust

# Apache JMeter – Overview



- "Apache JMeter" or just "JMeter" is a Java based open-source performance testing tool.
- The first version of JMeter was released back in 1998.
- JMeter allows the performance evaluation of a wide variety of protocols and technologies.
- JMeter supports a standard format for writing performance tests.

# Apache JMeter – Features

- As Apache JMeter has an open-source license, it is free to use and distribute.
- Apache JMeter has a simple, user-friendly and intuitive GUI for test development.
- Apache JMeter supports performance testing of different protocols and servers.
- As Apache JMeter is a Java based software, it is platform independent, portable and can run on Windows as well as Unix / Linux based operating systems.
- Apache JMeter has a multi-threaded framework which allows simulation of concurrent Virtual Users via threads and simultaneous functions via thread groups.
- Apache JMeter supports distributed testing with a controller instance distributing tests among multiple worker instances to generate a higher load on the target server.
- Apache JMeter can run tests in both GUI and Non-GUI mode.
- Apache JMeter is extensible and supports a variety of plugins for meeting different test requirements.
- Apache JMeter supports record and playback feature by which the user actions can be recorded using a browser.

# Apache JMeter – Types of Servers / Protocols

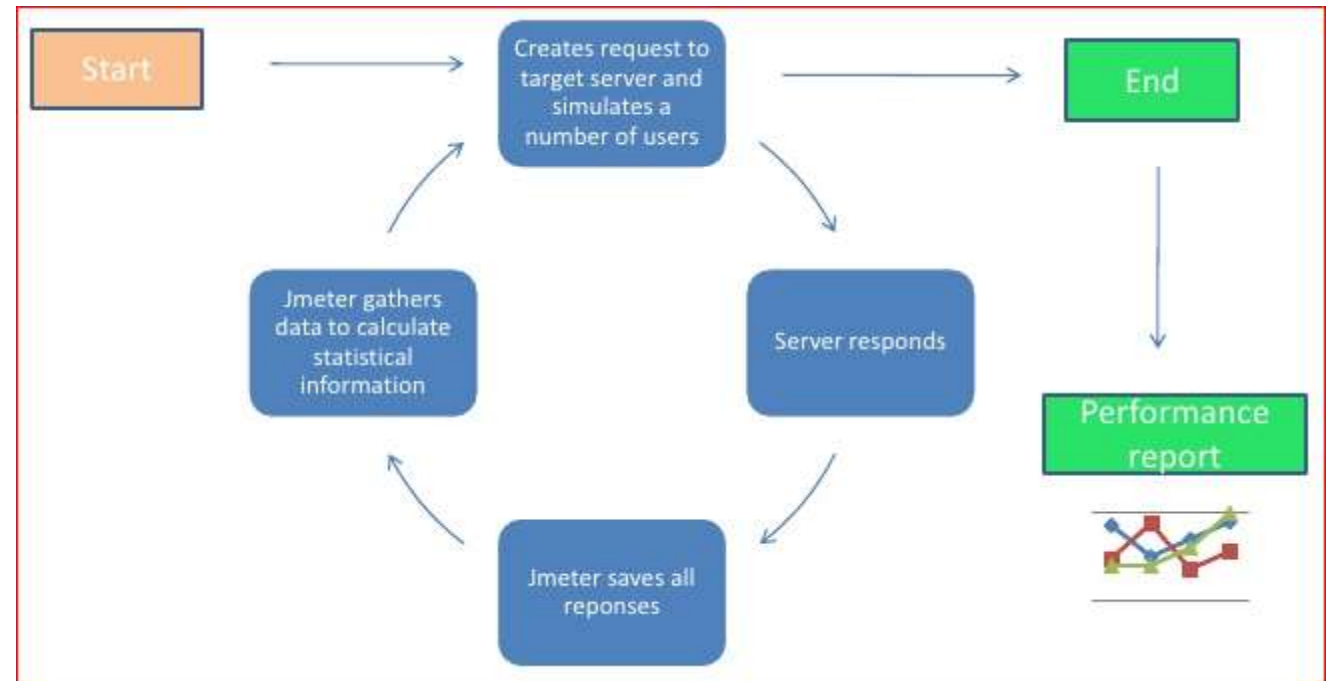
- JMeter natively supports the testing of the following Server Types / Protocols :

Server Type / Protocol	Description	Examples
HTTP, HTTPS	Protocol for communication between client and web server	Apache HTTP Server, NGINX, Microsoft IIS
REST, SOAP, XML-RPC	Protocol for exchanging structured data between webservices and networked applications	Axis2, WebSphere, GitHub API, Google Maps API
FTP	Protocol for transferring files between client and FTP server	vsftpd, ProFTPD, FileZilla Server
JDBC	Connection to databases	MySQL, Oracle DB, PostgreSQL
LDAP, LDAPS	Accessing and managing directory information	OpenLDAP, Microsoft AD
JMS	Connection to asynchronous messaging services	Apache ActiveMQ, RabbitMQ
SMTP, SMTP/TLS, SMTP/SSL	Connection to Mail Servers for sending emails	Gmail SMTP, MS Exchange Server, Postfix
POP3, IMAP, POP3S, IMAPS	Connection to Inbox Mail Servers	Gmail, MS Exchange
TCP and UDP	Connection oriented and connectionless communication	TelnetD, DNS, NTP

- JMeter can be configured to support additional protocols like WebSocket, Kafka etc. with plugins.

# Apache JMeter – How it Works ?

- Apache JMeter spawns a set of virtual users and sends the configured request to the target server.
- The target server receives the incoming request, processes the same and responds with the requested resource.
- JMeter receives the request, saves the responses and calculates the statistical information.
- At the end of the test, the virtual users are taken down and the performance report is generated.



# Apache JMeter – Limitations

- Running performance tests with JMeter can be resource intensive.
- JMeter's user interface and configuration options can be overwhelming for beginners, leading to a steep learning curve for new users.
- JMeter primarily operates at the protocol level and lacks the ability to simulate real browsers. JMeter does not support the execution of JavaScript, AJAX requests by default not does it render certain web elements like a browser
- JMeter provides very limited built-in real-time monitoring capabilities which makes the monitoring of the application's performance metrics during the test execution difficult.
- As test scripts grow in complexity, maintenance efforts can become challenging, especially when dealing with dynamic elements or frequent application updates.

# Apache JMeter – Installation

## Prerequisite –

- JMeter is a desktop application based on Java and runs in a JVM environment. Hence Java must be installed on the machine prior to running JMeter.

## Installing Java –

- Java can be installed by with the latest Java Runtime Environment (JRE) / Java Development Kit (JDK) [32 bit / 64 bit].
- Go to the following link : <https://www.oracle.com/java/technologies/downloads/>
- Download the JDK compatible with the Operating System.
- Run the downloaded executable and follow the onscreen instructions to install Java.

## Setting up JAVA\_HOME path variable –

### *On Windows :*

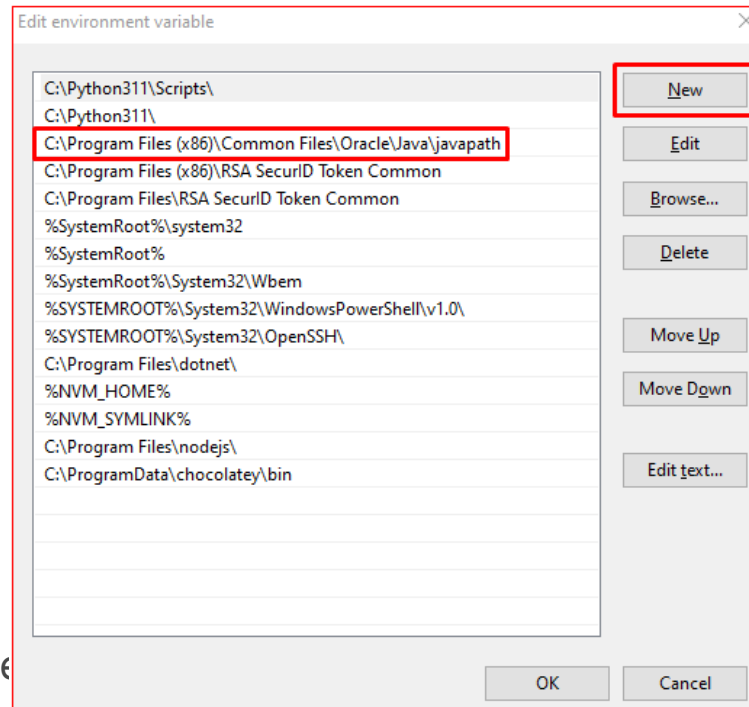
- Go to Control Panel → System And Security → System → Advanced System Settings → Environment Variables
- Verify whether Java Path is appended to the “Path” variable under System Variables. If it is not added by default, add it manually.



# Apache JMeter – Installation ...

For instance, say the Java Path is located @ ***C:\Program Files (x86)\Common Files\Oracle\Java\javapath***

This path can be appended to the existing System Variable for «Path» by selecting New as shown :



- Alternatively, the path can be set as a separate variable and then added to the "Path" system variable.

- To verify the Java installation run "java -version" in a command prompt / PowerShell and the installed Java version must be returned :

```
C:\>java -version
java version "1.8.0_371"
Java(TM) SE Runtime Environment (build 1.8.0_371-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.371-b11, mixed mode)
```

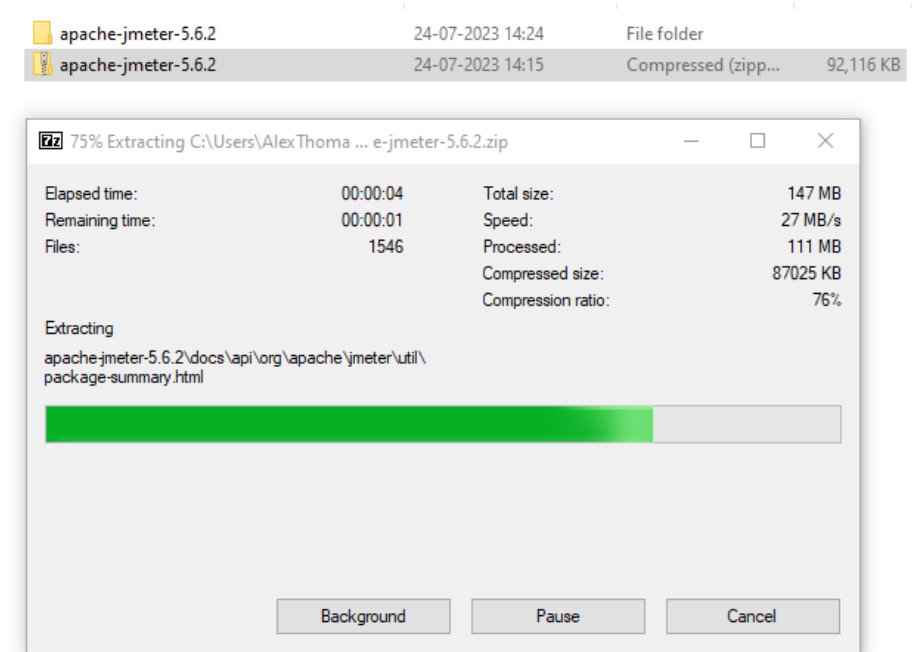
# Apache JMeter – Installation ...

## *On MacOS / other Unix based OS:*

- Open a terminal window.
- Execute "`export JAVA_HOME=<Path-of-Java-Home>`", replace the path with the actual path.
- Execute "`export PATH=$PATH:$JAVA_HOME`"
- Also set this in the shell profile settings based on the shell used (for e.g. `.bash_profile` for bash users / `.zshrc` for zsh users) in order to persist the settings.
- To verify the Java installation run "`java -version`" in the terminal.

## Installing Apache JMeter –

- JMeter comes as a bundled archive. To get the latest version (Version 5.6.2 at the time of making this video) visit :  
[https://jmeter.apache.org/download\\_jmeter.cgi](https://jmeter.apache.org/download_jmeter.cgi)
- Download the binary bundle as either .zip / .tgz whichever is preferred.
- Copy and extract the archive to any location as per choice.



# Apache JMeter – Installation ...

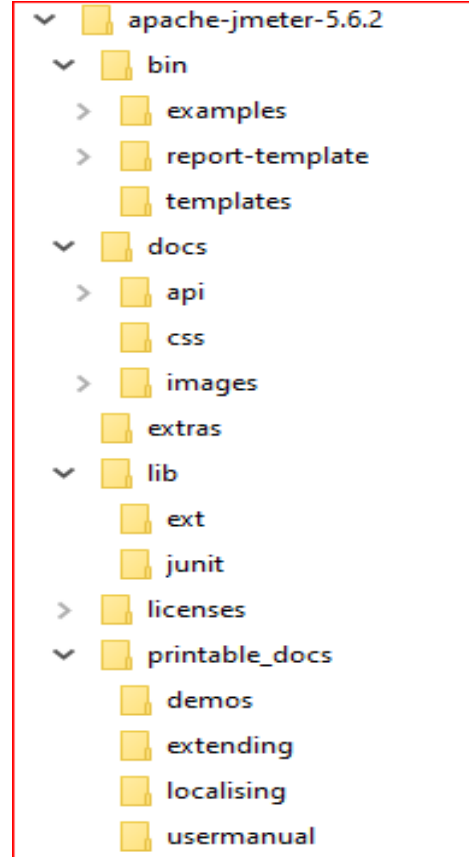
## Directories in extracted JMeter bundle –

- ***bin*** – This folder contains executable scripts to run and perform operations in JMeter.
- ***docs*** – This folder contains a well-documented user guide.
- ***extras*** – This folder contains miscellaneous items like samples illustrating the usage of Apache Ant, bean shell scripting etc.
- ***lib*** – This folder contains utility JAR files required by JMeter. Additional JARs may also be added.
- ***printable\_docs*** – This folder contains printable documentation for JMeter.

## Starting Apache JMeter –

JMeter can be started / launched in any of the following 3 methods :

- GUI Mode ➔ For launching JMeter as Client with GUI
- Server Mode ➔ For launching JMeter as Server for distributed testing
- Non-GUI Mode ➔ For launching JMeter as Client without a GUI in order to minimize GUI overhead



# Apache JMeter – Installation ...

## Starting JMeter in GUI Mode –

The bin folder holds all the executables and required for running / launching JMeter.

The startup of JMeter varies based on the OS on which the bundle is extracted :

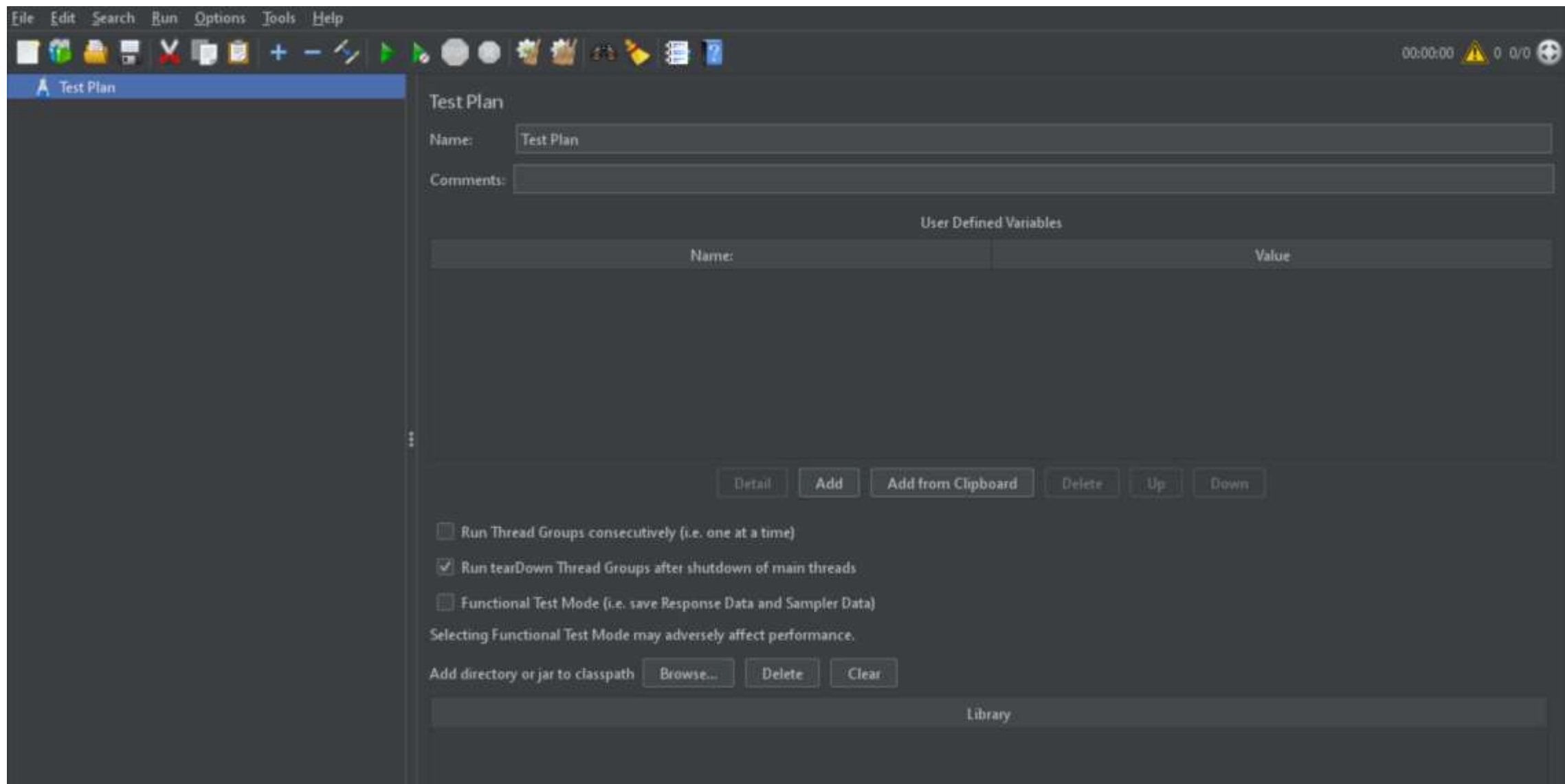
### *On Windows :*

➔ In order to launch JMeter on a Windows OS, execute the ***jmeter.bat*** present in the bin directory.

### *On MacOS / other Unix based OS :*

➔ In order to launch JMeter on a Mac OS / other operating systems which are Unix / Linux flavored, execute the ***jmeter.sh*** present in the bin directory.

# JMeter GUI



Q & A



# JMeter Test Plan

A Test Plan can be viewed as a container for running tests. It defines what to test and how to go about it. A complete test plan consists of one or more elements such as thread groups, logic controllers, sample-generating controllers, listeners, timers, assertions, and configuration elements. A test plan must have at least one thread group. We can store the frequently used variables in the test plan as user defined variables

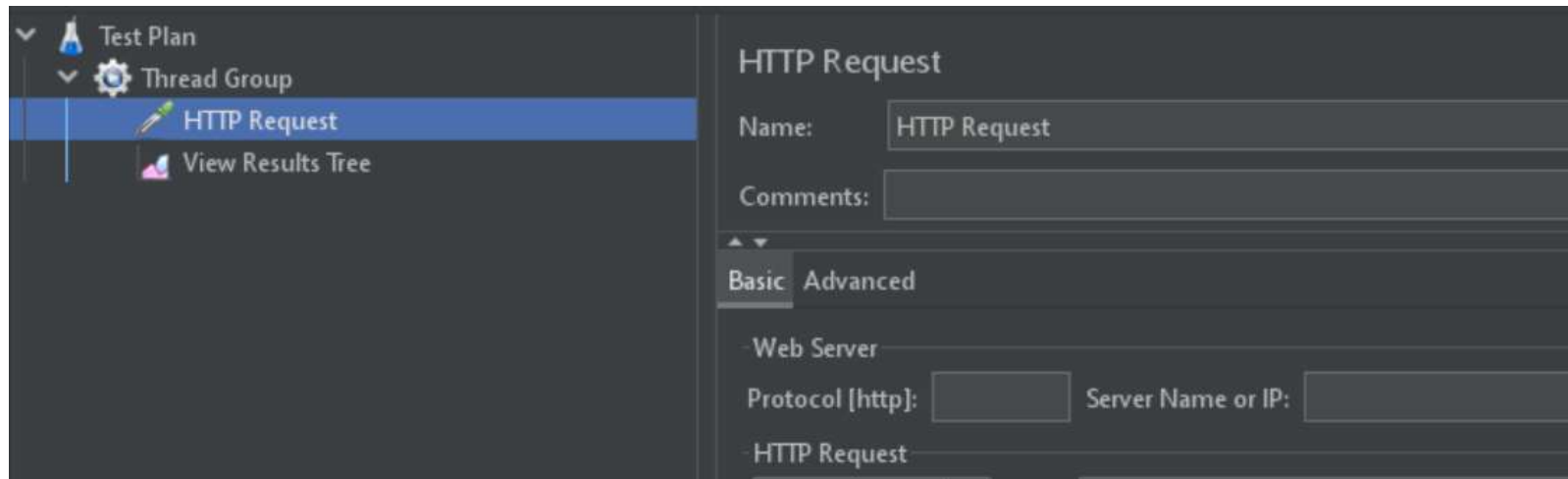
## Elements of Test Plan

- Thread Groups
- Samplers
- Logic Controllers
- Listeners
- Timers
- Assertions
- Config Element
- Pre-Processors
- Post Processors

# Test Plan

A simple test plan can be created by adding a thread group which is the first element and the below mentioned elements inside the thread group :

- HTTP Request sampler
- Listeners



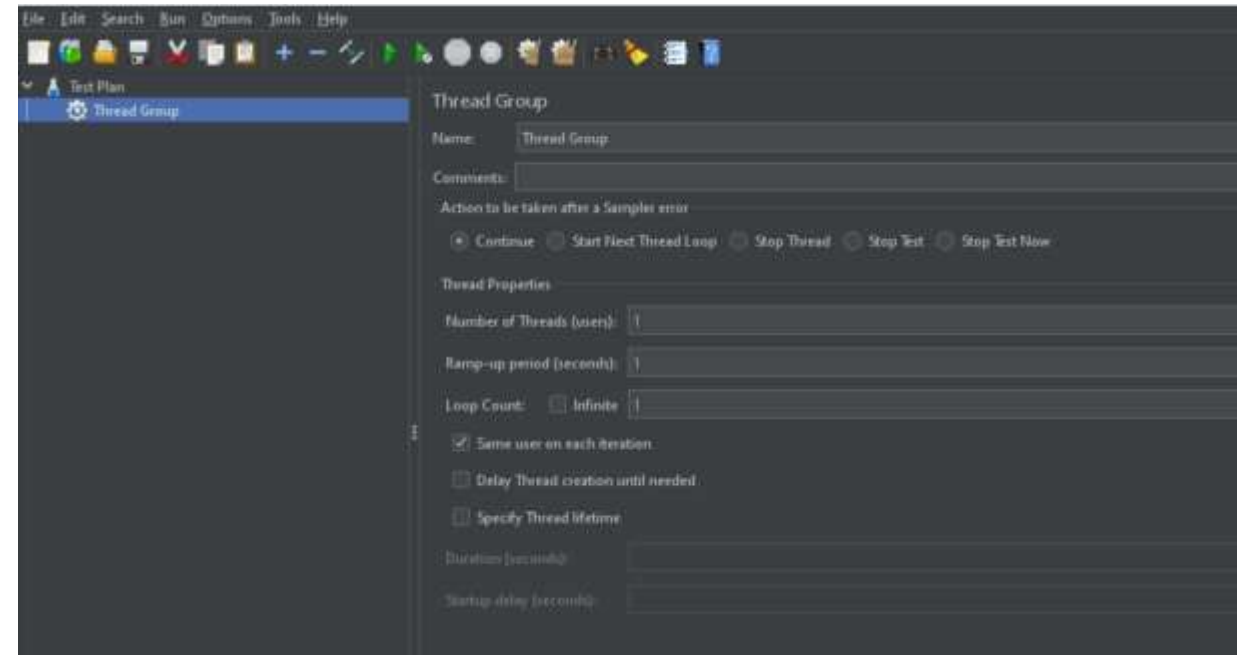


# Thread Group

- Thread group is the basic element of the JMeter Test plan.
- Thread group is a group of threads that are executing the same scenario.
- This is considered the beginning point of a test plan.
- Thread group holds other test elements like controllers, samplers, config elements, listeners etc.
- Each thread in the thread group will independently execute all the elements under the thread group while running the test plan.

## *How to add a thread group ?*

- Right-click the test plan.
- Click on **Add → Threads (Users) → Thread Group**



# Configuring Thread group

**Number of Threads:** This is the count of virtual users that we are expecting to connect to the server. For example, if we give 5, JMeter will simulate 5 virtual users that connect to the server and perform the same steps given. By default, it is set to 1 thread.

**Ramp-Up Period:** This can be defined as the time in which JMeter can bring the number of threads mentioned above into the running state. This is given in seconds. For example, If you set "Number of Threads" to "20", and "Ramp-Up Period to 40 seconds", then JMeter will wait till 40 seconds to make all threads up and running. By default, it is set to 1 second.

**Loop Count:** This indicates how many times each thread was supposed to perform the task. If the number of threads is 4 and the loop count is 2 then the same task will be performed 8 times. If this value is set to infinity the task will continue to run until the test was stopped. By default, value is one iteration.

# HTTP Request Defaults

- If we want to send multiple requests to the same web server, consider using an HTTP Request Defaults Configuration Element so we do not have to enter the same information for each HTTP Request.
- The main goals behind using the HTTP Default Request are to :
  - ➔ Avoid data duplication
  - ➔ Make test scripts more maintainable

# Samplers

Samplers allow JMeter to send specific types of requests to a server. They simulate a user request for a page from the target server. For example, you can add a HTTP Request sampler if you need to perform a POST, GET, PUT or DELETE on a HTTP endpoint.

***Few commonly used samplers are :***

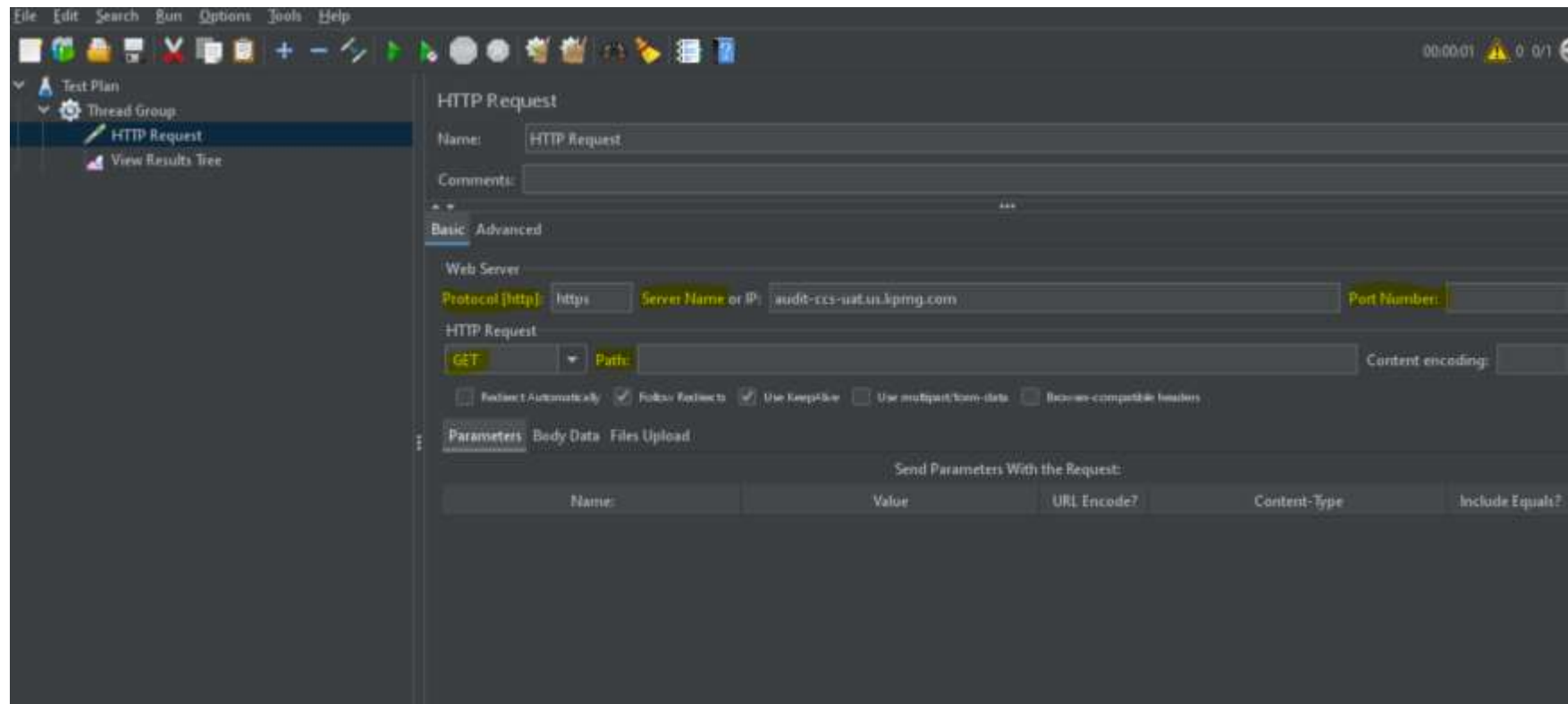
- HTTP Request
- FTP Request
- JDBC Request
- Java Request
- SOAP/XML Request
- RPC Requests

# Samplers

The HTTP request sampler can be added by:

Right click on **Thread Group** → **Add** → **Sampler** → **HTTP Request**

After adding Sampler, you can fill the details like Protocol, Server Name, Port, Path, Type of Request etc.



# Listeners

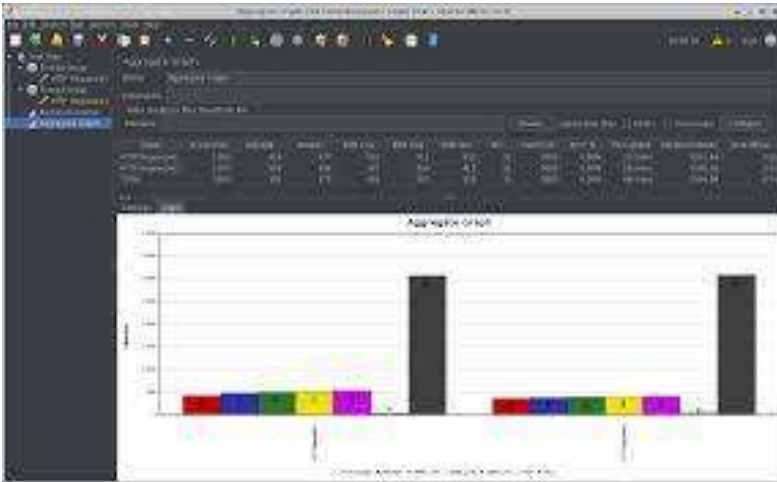
To view the execution results in detail various listeners may be used :

- Summary Report
- View Results Tree
- Aggregate Report
- Aggregate Graph

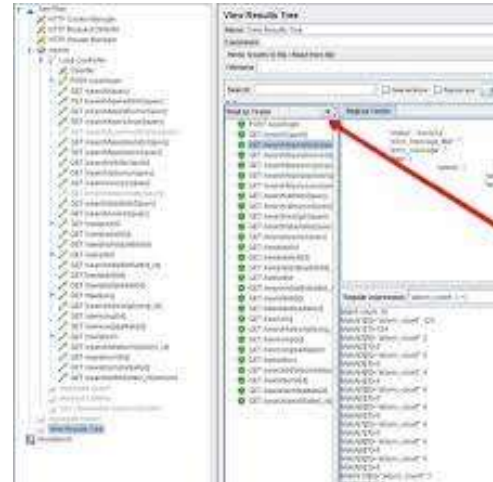
# Listeners

The listeners can be added by:

Right click on **Thread Group** → **Add** → **Listener** → **View Result Tree**



Aggregate Report



View Result Tree

The screenshot shows the 'Summary Report' window in JMeter. It displays a table of test results. The table has columns for Label, # Samples, Average, Min, Max, and Std. Dev. The data is summarized for various test elements, including 'View Result Tree', 'View Result Tree', 'View Result Tree', 'View Result Tree', 'View Result Tree', and 'TOTAL'.

Label	# Samples	Average	Min	Max	Std. Dev.
View Result Tree	8080	3304	0	32134	8
View Result Tree	8080	3304	0	32134	8
View Result Tree	8080	3304	0	32134	8
View Result Tree	8080	3304	0	32134	8
View Result Tree	8080	3304	0	32134	8
TOTAL	38560	7344	0	218752	8

Summary Report

# Logic controllers

It handles the order of processing Samplers/Requests in a Thread. Logic Controllers will decide “When & How” to send a request to a web server.

Some of the controllers Provided by JMeter:

- Once Only Controller
- If Controller
- Loop Controller
- Simple Controller
- Transaction Controller
- While Controller

Logic Controllers may be added by :

Right click **Thread Group** → **Select Add** → **Logic Controller**



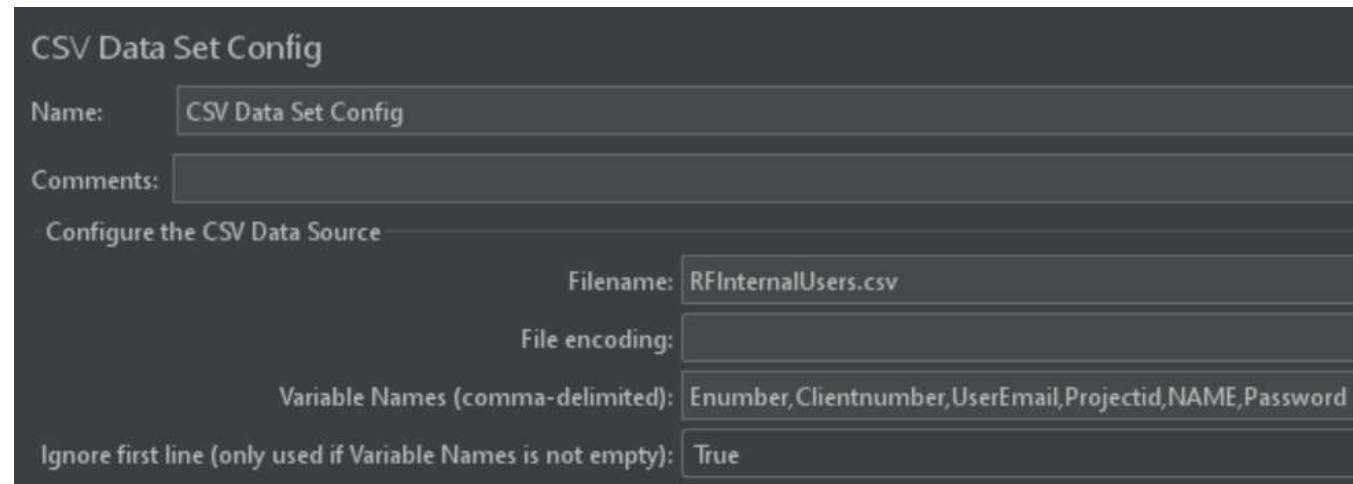
# Parameterization

**Parameterization** refers to the process of replacing hard-coded values in your test plan with parameters or variables. This allows you to create more flexible and dynamic test scenarios by varying input data during test execution.

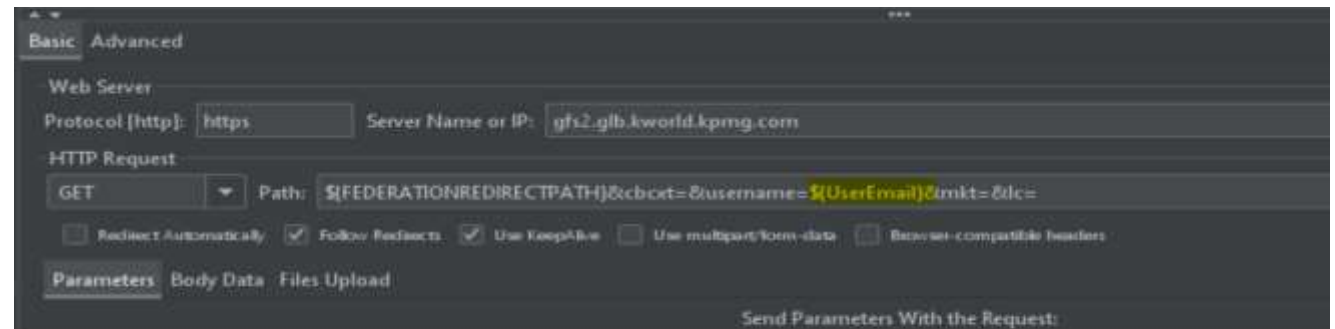
Few ways to achieve parameterization :

## → **CSV Data Set Config :**

- Use a CSV file to store your data (e.g., usernames, passwords, etc.).
- Configure the CSV Data Set Config element in your test plan to read data from the CSV file.
- Use variables in your test plan and reference them using the `${variable_name}` syntax.

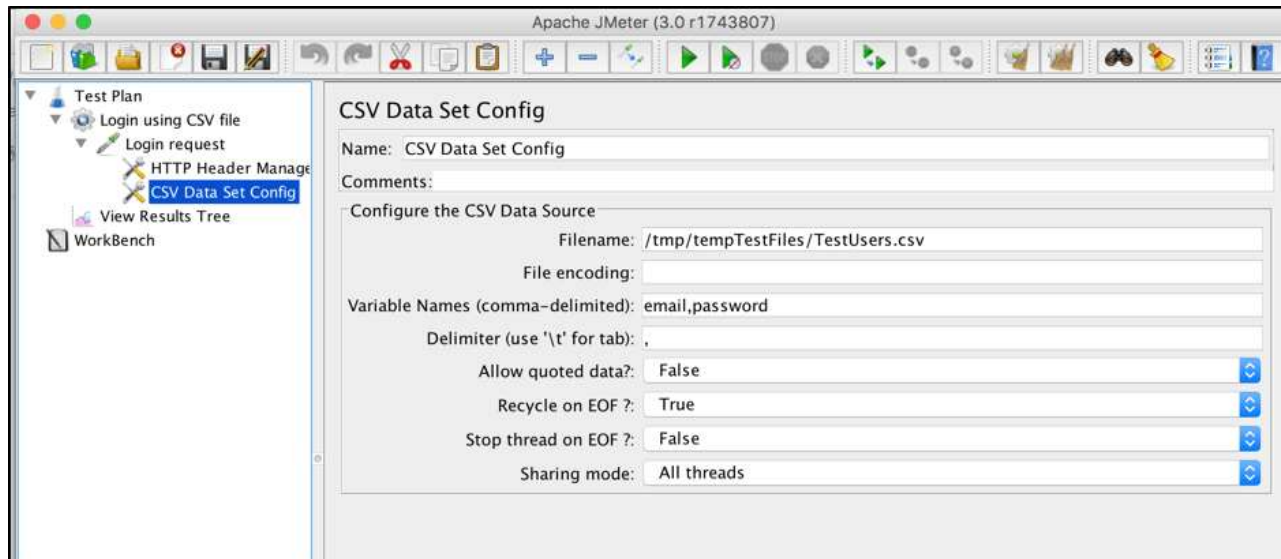


The screenshot shows the 'CSV Data Set Config' dialog box. It has a 'Name' field set to 'CSV Data Set Config' and an empty 'Comments' field. Below these is a section titled 'Configure the CSV Data Source'. It contains a 'Filename' field with 'RFInternalUsers.csv', an empty 'File encoding' field, a 'Variable Names (comma-delimited)' field with 'Enumber,Clientnumber,UserEmail,Projectid,NAME,Password', and an 'Ignore first line (only used if Variable Names is not empty)' checkbox that is checked and labeled 'True'.

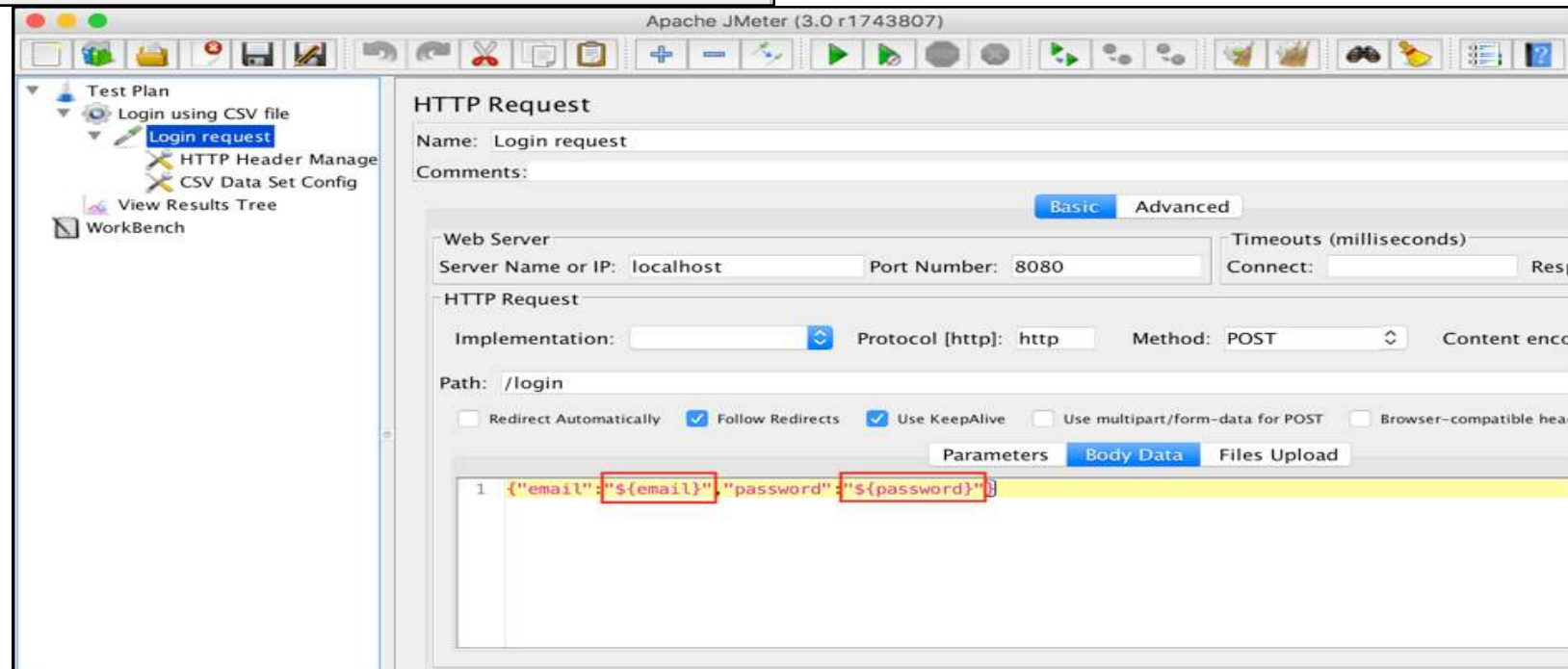


The screenshot shows the 'Basic' tab of the 'HTTP Request' sampler. It has tabs for 'Basic', 'Advanced', 'Parameters', 'Body Data', and 'Files Upload'. The 'Web Server' section shows 'Protocol [http]:' set to 'https' and 'Server Name or IP:' set to 'gfs2.glb.kworld.kpmg.com'. The 'HTTP Request' section shows the 'Method' set to 'GET' and the 'Path' set to '\$[FEDERATIONREDIRECTPATH]&cbset=&username=\${UserEmail}&link=&lc='. Below this are several checkboxes: 'Redirect Automatically' (unchecked), 'Follow Redirects' (checked), 'Use KeepAlive' (checked), 'Use multipart/form-data' (unchecked), and 'Browser-compatible headers' (unchecked). At the bottom, there is a 'Send Parameters With the Request:' label.

# Parameterization in JMeter



```
tempTestFiles cat /tmp/tempTestFiles/TestUsers.csv
testuser1@test.com,password1
testuser2@gmail.com,password2
testuser3@gmail.com,password3
testuser4@gmail.com,password4
testuser5@gmail.com,password5
testuser6@gmail.com,password6
testuser7@gmail.com,password7
testuser8@gmail.com,password8
testuser9@gmail.com,password9
testuser10@gmail.com,password10
```

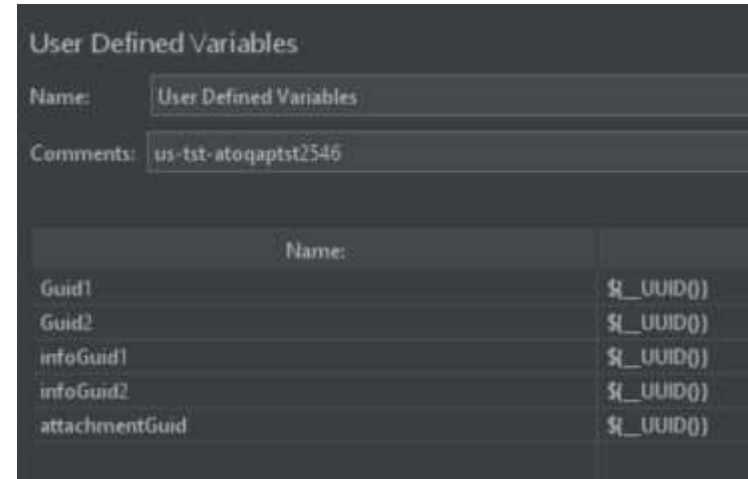


# Parameterization ...

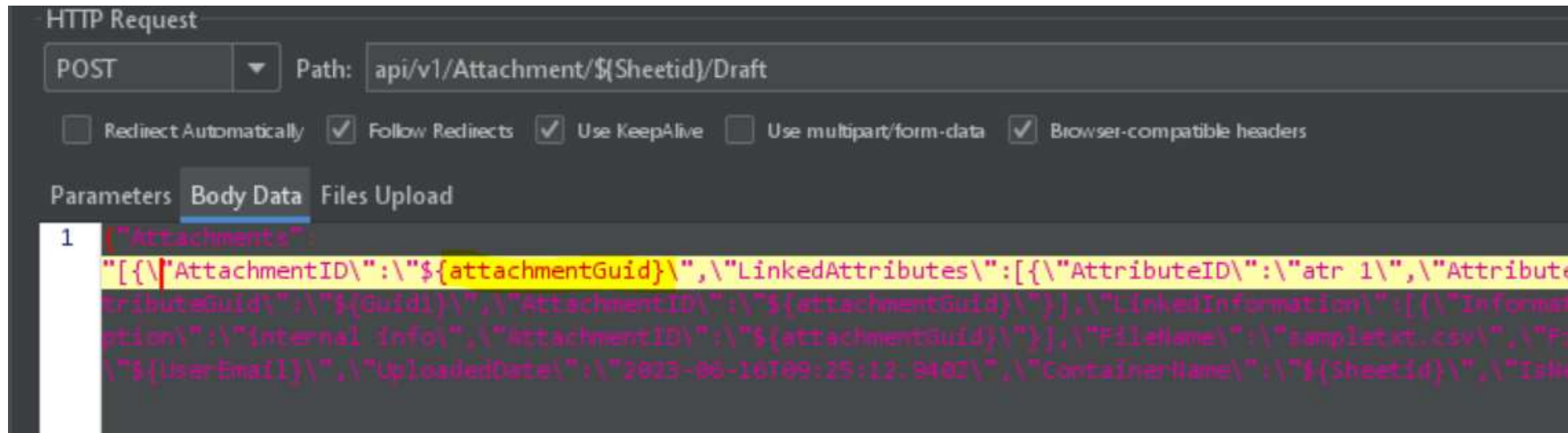
→ *User Defined Variables:*

*Define variables in the User Defined Variables configuration element.*

*Reference these variables in your test plan using the `${variable_name}`*



Name:	
Guid1	\${_UUID()}
Guid2	\${_UUID()}
infoGuid1	\${_UUID()}
infoGuid2	\${_UUID()}
attachmentGuid	\${_UUID()}



HTTP Request

POST Path: `api/v1/Attachment/${Sheetid}/Draft`

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

Parameters Body Data Files Upload

```
1 [{"Attachments": [{"AttachmentID": "${attachmentGuid}", "LinkedAttributes": [{"AttributeID": "atr_1", "AttributeGuid": "${Guid1}", "AttachmentID": "${attachmentGuid}"}], "LinkedInformation": [{"InformationID": "internal_info", "AttachmentID": "${attachmentGuid}"}], "FileName": "samplext.csv", "FileSize": 1024, "UserEmail": "test@example.com", "UploadDate": "2023-06-16T09:25:12.940Z", "ContainerName": "${Sheetid}", "IsHidden": false}]}]
```

# Pre-Processors & Post-Processors

Processors are of two types – ***Pre-Processors*** and ***Post-Processors***.

## ***Pre-Processors :***

Executes before the sampler execution.

Pre-Processors can be used for different performance testing needs, like fetching data from a database, setting a timeout between sampler execution or before test data generation.

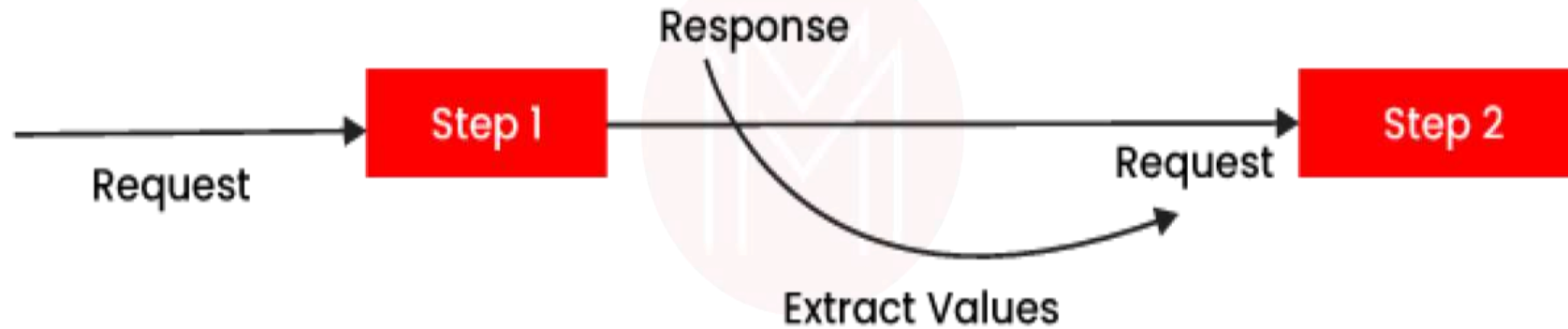
*For example : User Parameters, JSR223 Pre-Processors*

## ***Post-Processors :***

Used to extract the response data from the server and to save the specific extracted values for later use.

*For example : Regular Expression Extractor, JSON Extractor, JSR223 Post-processor*

# Correlation



## ***JSON Extractor:***

Use the JSON Extractor to extract values from JSON responses and store them in variables for later use.

## ***Regular Expression Extractor***

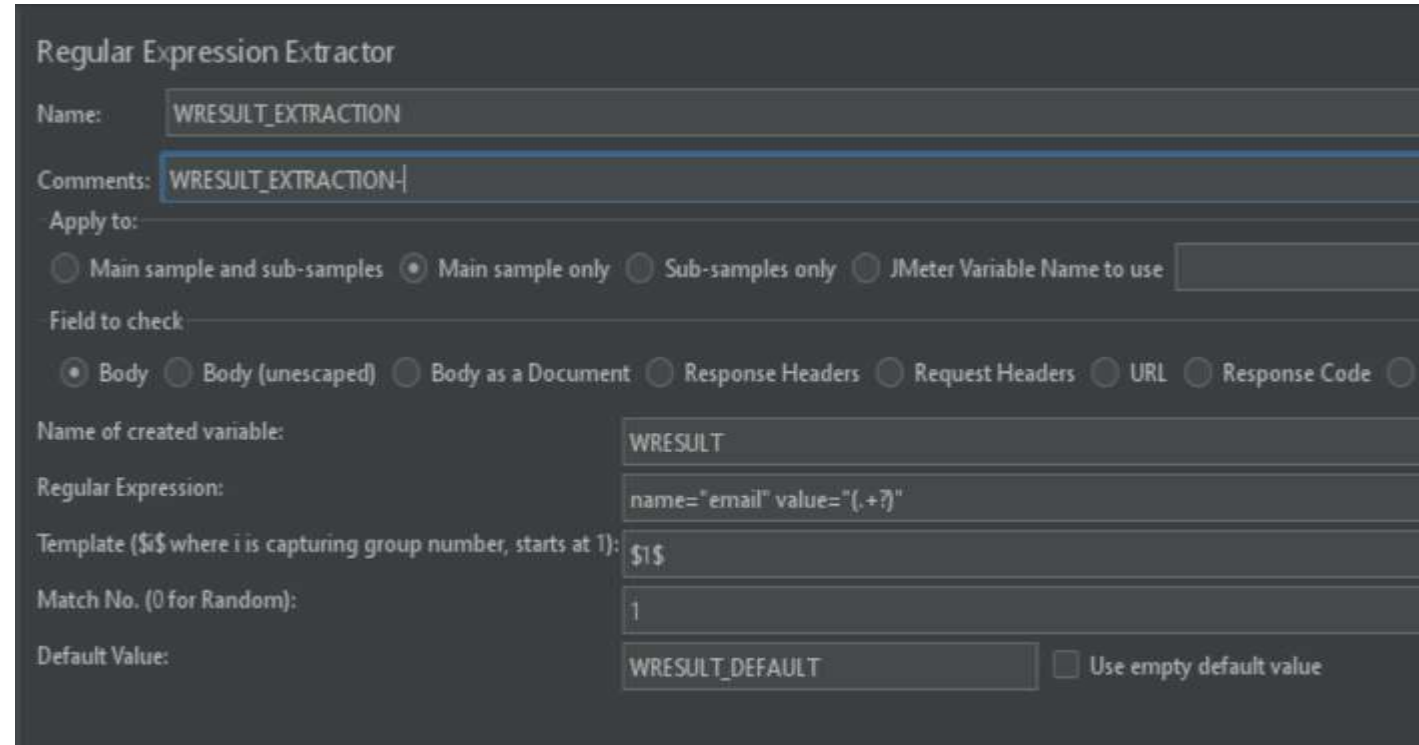
Regular Expression Extractor allows you to extract values using regular expressions.

## ***Boundary Extractor***

Boundary Extractor extracts data using left and right boundaries of the dynamic value which needs to be captured

# Regular Expression Extractor

- Regular expressions are a tool used to extract a required part of the text by using advanced manipulations.
- In JMeter, the Regular Expression Extractor is useful for extracting information from the response.
- For example, if the response has "name='email' value='xyz@abc.com'" and you need the email value for the next request, then the RegEx component will be using for extraction with a regular expression of name='email' value="(.+?)".

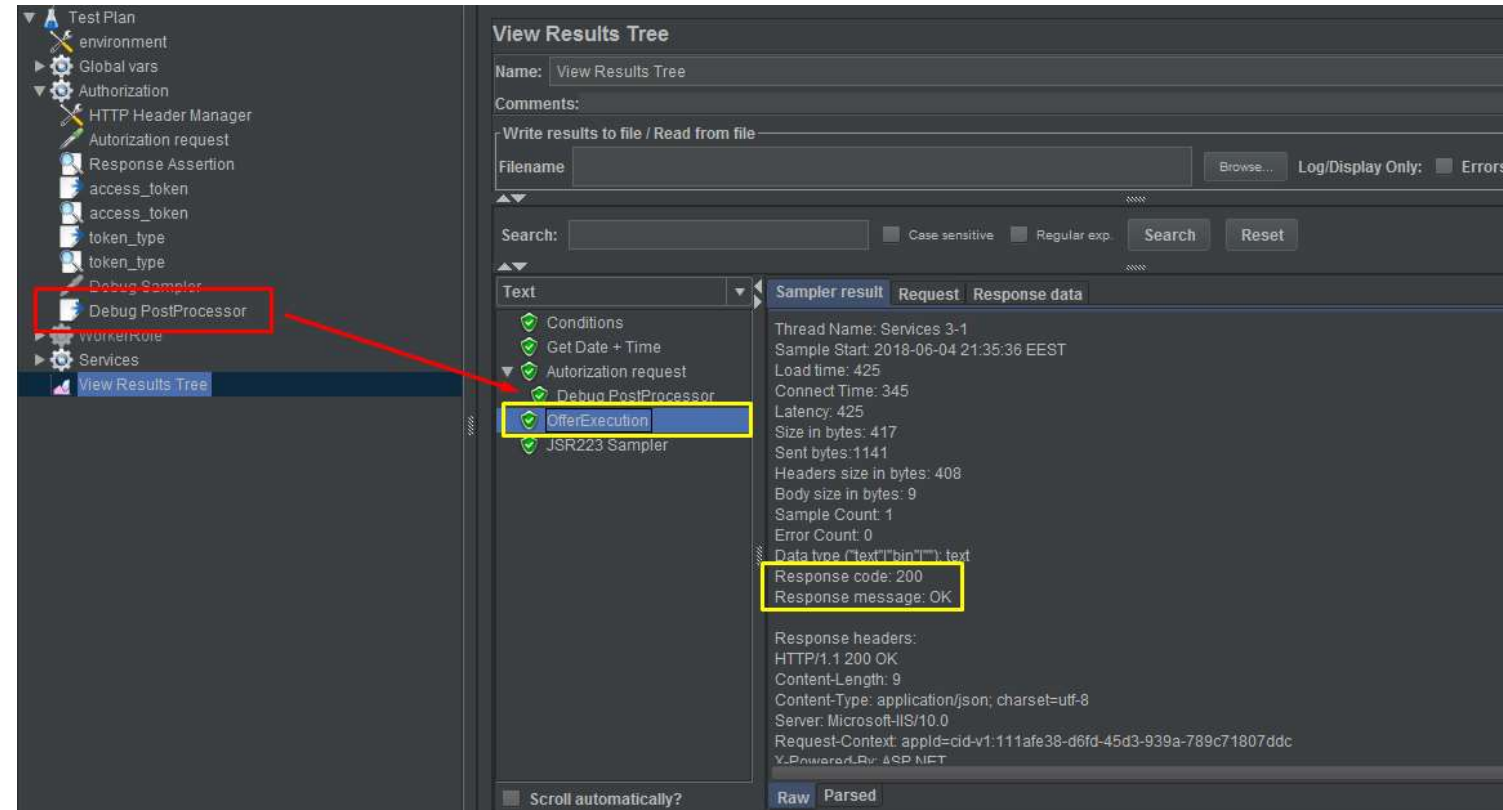


The screenshot shows the 'Regular Expression Extractor' configuration window in JMeter. The 'Name' field is set to 'WRESULT\_EXTRACTION'. The 'Comments' field contains 'WRESULT\_EXTRACTION-'. Under 'Apply to:', the 'Main sample only' radio button is selected. Under 'Field to check:', the 'Body' radio button is selected. The 'Name of created variable:' field is 'WRESULT'. The 'Regular Expression:' field contains 'name="email" value="(.+?)"'. The 'Template (\$i\$ where i is capturing group number, starts at 1):' field contains '\$1\$'. The 'Match No. (0 for Random):' field contains '1'. The 'Default Value:' field is 'WRESULT\_DEFAULT', and the 'Use empty default value' checkbox is unchecked.

Regular Expression Extractor	
Name:	WRESULT_EXTRACTION
Comments:	WRESULT_EXTRACTION-
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
Field to check:	<input checked="" type="radio"/> Body <input type="radio"/> Body (unescaped) <input type="radio"/> Body as a Document <input type="radio"/> Response Headers <input type="radio"/> Request Headers <input type="radio"/> URL <input type="radio"/> Response Code
Name of created variable:	WRESULT
Regular Expression:	name="email" value="(.+?)"
Template (\$i\$ where i is capturing group number, starts at 1):	\$1\$
Match No. (0 for Random):	1
Default Value:	WRESULT_DEFAULT <input type="checkbox"/> Use empty default value

# Debug Sampler

- The debug sampler is a component which help to see the variable values in the "View Results Tree" component (if added to the test plan).
- It helps in troubleshoot your script variables. The sampler spits out all variable names and values including arrays
- The debug features do not work if you run the JMeter scripts using the CLI. The debugging features work only in GUI mode.



# Assertions

Assertions are used to verify and validate the data of requests that we have sent to the server.

Using assertions, we can compare the expected result with the actual result.

Few commonly used assertion elements are :

## ***Response Assertion :***

Response assertions are used to verify the patterns in the response body received from the server. Different pattern matching rules like contains, matches, equal, etc. can be used to verify the response.

## ***Size Assertion :***

Size assertion is used to verify the expected number of bytes. We can add the expected size in bytes and verify with the different types of comparisons like >, <, =, etc.

## ***JSON Assertion:***

It is used when we are getting JSON response. It will be mostly when we test rest API.

Assertions can be added by :

Right click on **Thread Group** → **Add** → **Assertion**



# Assertions

Here the assertion has given for the response code in a way that if the code is 200 or 404 then the status will be ignored.

### Response Assertion

Name:

Response Assertion

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|404)

# Timers

- JMeter requests / actions (across threads) usually run one after another without any time delay.
- This can be unrealistic in nature because a normal user will take time gaps to read or think between performing each action.
- This is called Think Time, and it should be added to make the script more realistic and reliable.
- To perform such operations JMeter provides several timers. Depending on the purpose, the user can select the one which suits their scenario.

Few commonly used timers include :

➔ ***Constant Timer*** : This can be used to add a constant delay / think time between each request.

➔ ***Uniform Random Timer*** : Uniform Random can be used to add a random amount of think time between the user requests.

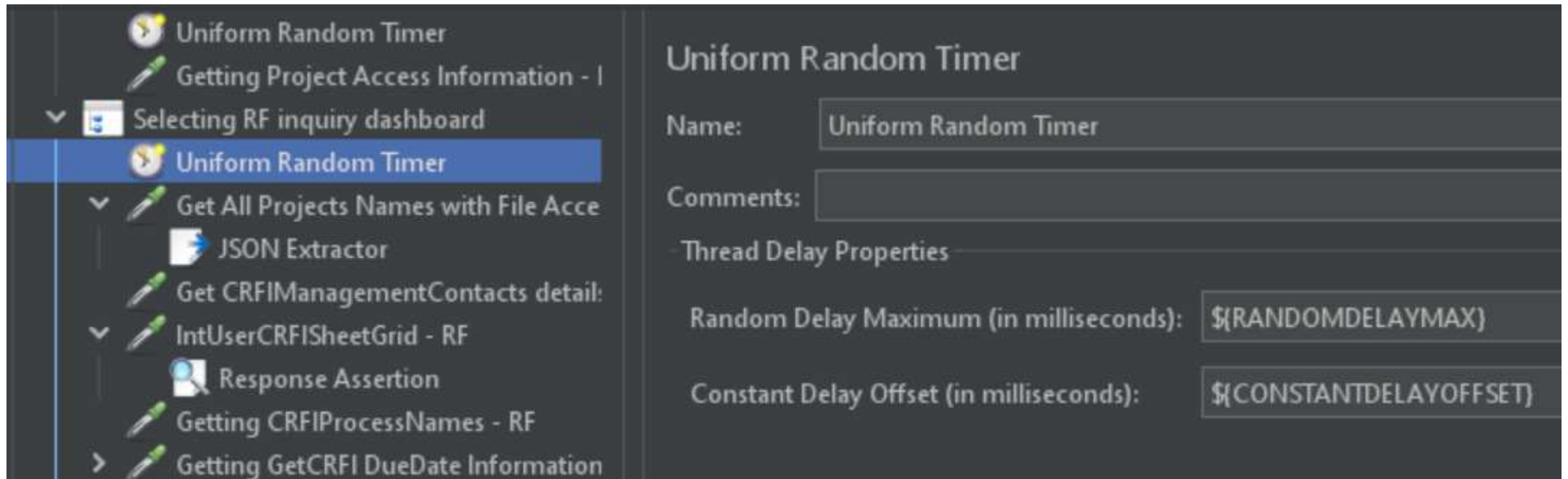
➔ ***Synchronizing Timer*** : This time is used to hold the threads until X number of threads have arrived, and then they are all released at once.

The Timers can be added by :

**Right click on Thread group → Add → Timer.**

# Timers ...

Here the uniform random timer is used and the maximum delay or think time should be given as random delay, then it will give a random think time to user requests which are not above the given value.



The screenshot displays the JMeter GUI. On the left, a tree view shows the test plan structure. The 'Uniform Random Timer' is selected under the 'Selecting RF inquiry dashboard' folder. The right pane shows the configuration for the 'Uniform Random Timer'.

**Uniform Random Timer**

Name:

Comments:

Thread Delay Properties

Random Delay Maximum (in milliseconds):

Constant Delay Offset (in milliseconds):

Q & A



# Running tests in JMeter

## ***GUI mode:***

After creating a test plan with necessary components like samplers, timers, assertions, extractors, listeners etc. adding to a thread group and saving it to a location, user can click on the green play button to run the test in the GUI mode.

## ***CLI mode:***

Open a command prompt or terminal and navigate to the "bin" directory within your JMeter installation.

Use the following command to run JMeter in CLI mode:

```
./jmeter -n -t /path/to/your/testplan.jmx -l /path/to/results.jtl -e -o /path/to/report
```

-n : This option specifies JMeter to run in non-GUI mode.

-l : Specifies the path to save the results in JTL format.

-t : Specifies the path to the JMeter test plan file.

-J : Define additional properties. For example, -JUsers=100 -JDuration=300

# View Results Tree

## ***View Results Tree:***

- Each sample (HTTP request, JDBC request, etc.) executed during the test will be listed in the "View Results Tree."
- Select a specific sample to see detailed information about the request and response.
- Look for the response code to check if the request was successful (e.g., 200 for HTTP).
- Inspect the response data to verify if the server's response is as expected.

## ***Tabs in View Results Tree:***

- Request: Displays information about the request, including method, URL, parameters, headers, etc.
- Response Data: Shows the raw response data received from the server.
- Response Headers: Displays the headers sent by the server in the response.
- Request Headers: Shows the headers sent by JMeter in the request.
- Sampler Result: Provides additional details about the sampler result, including response code, response message, etc.

# View Result Tree ...

The screenshot shows the 'View Results Tree' window in a testing tool. The left sidebar contains a tree view of the test plan, with 'View Results Tree' selected. The main area displays the results of the selected test step, 'ADFS\_LS'. The results are organized into a table with columns for 'Text', 'Sampler result', 'Request', and 'Response data'. The 'Text' column shows the test steps, and the 'Response data' column shows the response details for the selected step.

**View Results Tree**

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only:

Search: Case sensitive Regular exp. Search Reset

Text	Sampler result	Request	Response data
LoginsLoad			
Authorize			
AuthorizeSSOReload			
GetCredentialType			
ADFS_LS			
ADFS_LS_PULLSTATUS=0			
ADFS_LS_WIA_PULLSTATUS=0			
LoginSRF			
Token Attachment User			
CCS AUTHORIZATION			

Thread Name: Attachments User Flow 1-1  
Sample Start: 2023-11-17 12:48:16 IST  
Load time: 447  
Connect Time: 273  
Latency: 445  
Size in bytes: 34270  
Sent bytes: 2583  
Headers size in bytes: 238  
Body size in bytes: 34032  
Sample Count: 1  
Error Count: 0  
Data type ("text"|"bin"|""): text  
Response code: 200  
Response message: OK

HTTPSampleResult fields:  
ContentType: text/html; charset=utf-8  
DataEncoding: utf-8

Scroll automatically? Raw Parsed

# Summary Report

## Summary Report Listener

The "Summary Report" listener in JMeter provides aggregated information about the test. Summary Report will be showing as a table.

### Columns in Summary Report:

- ***Label*** : The name of the sampler or controller.
- ***# Samples*** : The total number of samples executed.
- ***Average*** : This indicates the average time taken for the requests to be processed by the server. A higher average response time may indicate performance issues.
- ***Median*** : The median response time.
- ***90% Line*** : The response time below which 90% of the samples fall.
- ***Min*** : The minimum response time observed.
- ***Max*** : The maximum response time observed.
- ***Error %*** : This shows the percentage of requests that resulted in errors. A higher error percentage may indicate issues with the application or server.



# Summary Report ...

File Edit Search Run Options Tools Help

00:01:18 0 0/1

Test Plan

- Error Logger
- Internal User Flow
- External User Flow
- RF - Internal User Flow
- RF - External User Flow
- Attachments User Flow
  - HTTP Cookie Manager
  - HTTP Cache Manager
  - CSV Data Set Config
  - Random Variable
  - User Defined Variables
  - CCS AUTHORIZATION
  - Attachments User Flow
    - Aggregate Report
    - View Results Tree
    - Debug Sampler
    - Save Responses to a file
    - Summary Report
    - View Results Tree
    - Summary Report

### Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename:   Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB...	Sent KB/sec	Avg. Bytes
LoginLoad	1	752	752	752	0.00	0.00%	1.3/sec	3.37	0.52	2595.0
Authorize	1	222	222	222	0.00	0.00%	4.5/sec	41.95	4.54	9536.0
AuthorizeSS...	1	149	149	149	0.00	0.00%	6.7/sec	116.79	10.67	17819.0
GetCredenti...	1	238	238	238	0.00	0.00%	4.2/sec	17.98	15.01	4381.0
ADFS_LS	1	447	447	447	0.00	0.00%	2.2/sec	74.67	5.64	34270.0
ADFS_LS_PU...	1	101	101	101	0.00	0.00%	9.9/sec	12.68	25.10	1311.0
ADFS_LS_WI...	1	615	615	615	0.00	0.00%	1.6/sec	18.35	5.06	11553.0
LoginSRF	1	598	598	598	0.00	0.00%	1.7/sec	15.16	17.84	9286.0
Token Attac...	1	270	270	270	0.00	0.00%	3.7/sec	23.09	18.13	6383.0
CCS AUTHO...	1	3392	3392	3392	0.00	0.00%	46.4/hour	1.22	0.39	97134.0
TOTAL	10	678	101	3392	928.21	0.00%	7.7/min	2.45	0.78	19426.8

# HTML Dashboard Report

Using the below code format user can generate HTML report in JMeter in CLI (Command-Line Interface) mode.

```
./jmeter -n -t /path/to/your_test_plan.jmx -e -o /path/to/report_output_folder
```

- -n: Run JMeter in non-GUI (CLI) mode.
- -e: Generate the HTML report.
- -o: Specify the path to the output folder where the HTML report will be saved.
- -t: Specify the path to your JMeter test plan file.

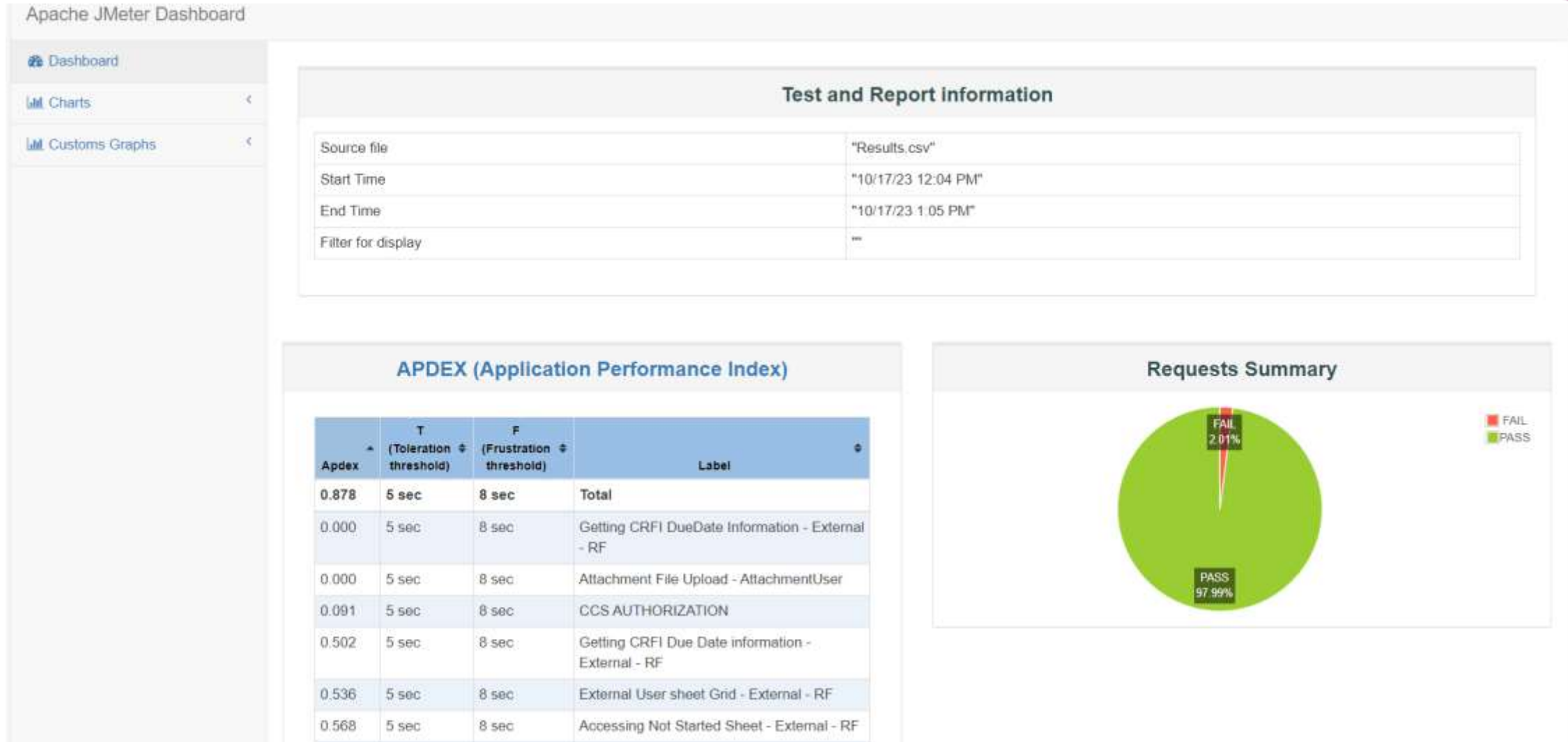
After the test execution is completed, an HTML report will be generated in the specified output folder.

This HTML report will include various charts and statistics summarizing the results of your test. Open the index.html file in a web browser to view the report.

## ***Note :***

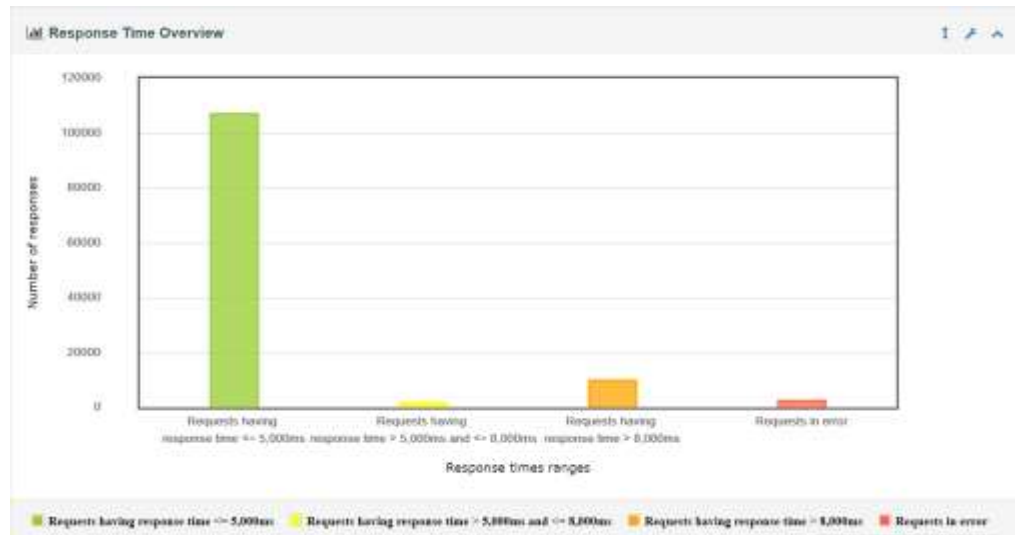
Replace /path/to/your\_test\_plan.jmx with the actual path to the JMeter test plan file and /path/to/report\_output\_folder with the desired output folder for the HTML report.

# HTML Dashboard Report ...



# HTML Dashboard Report ...

Different types of charts are available in the HTML report like Response Time Overview Graph, Hits per second etc. to analyze the report further.



Response Time Overview



Hits per second

# Error Details in HTML report

Errors			
Type of error	Number of errors	% in errors	% in all samples
400/Bad Request	1896	78.22%	1.57%
500/Internal Server Error	508	20.96%	0.42%
504/Gateway Time-out	6	0.25%	0.00%
403/Forbidden	6	0.25%	0.00%
Non HTTP response code: java.net.SocketException/Non HTTP response message: Connection reset by peer: socket write error	5	0.21%	0.00%
404/Not Found	3	0.12%	0.00%

The errors during execution will be listed in the “Error” section of the report. Error code with their number of occurrence and % are listed.

# Error Details in HTML report ...

- 5.X.X. HTTP error codes (500, 502, 503, 504, etc.): Each of them specifies a different problem but the common denominator they share is that something is wrong with the website's server. It shows the server encountered an unexpected condition that prevented it from fulfilling the request. This should be reported after the analysis.
- 4.X.X HTTP error codes(400,401,403): This indicates the server cannot or will not process the request due to something that is perceived to be a client error (for example, malformed request syntax, invalid request message framing, or deceptive request routing).
- **Note :** After a 500 error the depended request will not receive the required variable which result in the 400 error of the following request.

Q & A



Thank you!

