# Performance Testing Documentation

## Objective

The objective of this performance testing is to create and execute a script for the API https://www.freetogame.com/api/game?id=455 with data for ID 450 to 455. The script includes all parameters, response assertions, response extractions, and follows proper naming conventions.

## Approach

The approach selected for this task is to use JMeter, an open-source software designed to load test functional behavior and measure performance. JMeter allows us to simulate multiple users with concurrent threads, create a heavy load against the web application under test, and analyze overall performance under different types of load.

## Instructions for Creating the Script

1. Open JMeter in your system.
2. Create a new Test Plan:
   1. Right-click on the Test Plan and add a new Thread Group.
   2. In Thread Properties,
      1. Set Number of Threads (users): ${\_\_P(threads,1)}
      2. Set Ramp-up period (seconds): ${\_\_P(rampup,1)}
      3. Check the Specify Thread lifetime and set Duration (seconds): ${\_\_P(duration,300)}
3. Add HTTP Request Sampler:
   1. Right-click on the Thread Group, go to Add -> Sampler -> HTTP Request.
   2. In the HTTP Request, set the server name or IP to [www.freetogame.com](http://www.freetogame.com).
   3. Set the path to /api/game.
   4. Set the method to GET.
4. Add CSV Data Set Config:
   1. Right-click on the Thread Group, go to Add -> Config Element -> CSV Data Set Config.
   2. Set the filename to the location of your CSV file containing the IDs 450 to 455. (C:/Users/rohit/Desktop/apache-jmeter-5.6.2/bin/csv\_files/game\_ids\_450\_to\_455.csv)
   3. Set the variable name to “game\_id”.
5. Parameterize the HTTP Request:
   1. In the HTTP Request, under the parameters tab, add a new row with the name as id and the value as ${game\_id}.
6. Add JSON Extractor:
   1. Right-click on the HTTP Request, go to Add -> Post Processors -> JSON Extractor.
   2. Set Name of Created variables: response\_game\_id   
      JSON Path expressions: $.id   
      Match No: 1
7. Add Response Assertion:
   1. Right-click on the HTTP Request, go to Add -> Assertions -> Response Assertion.
   2. Apply to JMeter Variable Name to response\_game\_id
   3. Select Field to Test to Text Response  
      Pattern Matching Rules to Contains  
      Pattern to test to ${game\_id}
8. Add Constant Throughout Timer(For Test 3):
   1. Right-click on the Thread Group -> Add -> Timer -> Constant Throughput Timer
   2. In Delay before each affected sampler, set Target Throughput(in samples per minute) : ${\_\_P(throughput,7)}
   3. We disable it for Test 1 and Test 2

## Instructions for Executing the Script

1. We ensure Java, JMeter are in the path.
2. Open Command Line Interface

Execute the commands as follows for test specified:  
 **Test 1** : jmeter -n -t game\_id\_api\_test.jmx -Jthreads=1 -Jduration=300 -l results\_1user\_15\_12\_2023.jtl  
 **Test 2** : jmeter -n -t game\_id\_api\_test.jmx -Jthreads=5 -Jrampup=25 -Jduration=300 -l results\_5user\_15\_12\_2023.jtl

**Test 3** : jmeter -n -t game\_id\_api\_test.jmx -Jthreads=5 -Jrampup=25 -Jduration=300 -Jthroughput=7 -l results\_5user\_7ThrougputPerMin\_15\_12\_2023.jtl

## Test Execution

The script was executed under the following conditions:

**Test 1**: The script was executed for 1 user for a duration of 5 minutes.

**Test 2**: The script was executed with 5 users for a duration of 5 minutes, with each user ramping up in 5 seconds.

**Test 3**: The script was executed with 5 users for a duration of 5 minutes, with each user ramping up in 5 seconds to achieve 7 constant throughput.

## Results

The results of the test executions are saved in the [results.jtl](https://github.com/rHohith/Perfomance-API-Testing) files. These files contains the details of all the requests made during the test, including any errors that occurred.

## Observations and Recommendations

* **Observations**: The API was able to handle the load of 1 user and 5 users without any significant increase in response time or error rate. However, there were a few timeouts during the peak load, indicating that the server might be struggling to handle the load.
* **Recommendations**: It's recommended to investigate the timeouts and see if there's any room for optimization on the server side. Also, it would be beneficial to conduct further testing with higher loads to identify the maximum capacity of the server.

## Glossary:

Resources (Github): [Script and Result Files](https://github.com/rHohith/Perfomance-API-Testing)