***Non Functional Test Plan***

*Table of Contents*

1. *Introduction*
2. *Environment Setup*
3. *Test Scenarios*
   1. *Default Test*
   2. *Load Test*
   3. *Stress Test*
   4. *Spike Test*
   5. *Endurance Test*
4. *Automation*
5. *To-Do*
6. *References*

*Introduction*

The purpose of this document is to explain in detail all use cases for non functional tests which are required to benchmark and capture the KPI for services and system. Non functional tests to be tested with this document includes system performance tests only. Other non functional tests like security and network tuning is not included in this version.

*Environment*

Imitial test system is configured according to recommendations on <https://support.smartbear.com/readyapi/docs/loadui/intro/system-requirements.html>. Test system will be tuned and documented based on results found during testing. An assumption is made to use 4 different types test systems

* Subset of production system with fewer servers of lower specification
* Subset of production system with fewer servers of the same specification
* Replica of productions system
* Actual production system

**Test Scenarios**

**Default Test:**

**These tests are done to record data that can be used for benchmarking. Wait of 5mins is introduced assuming the think time for calling the API will be 5mins. Data for the API call with and without the think time should be the same in ideal case, but will server as a good indicator to calculate average. All tests are executed by a single thread, unless specified otherwise.**

*DT1: numSeatsAvailable API Cases*

*Quick recap: Call to numSeatsAvailable API should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| DT1-1 | A call is made to *numSeatsAvailable* API | Load time < 100ms  Size < 100bytes  Response code = 200  Latency < 400ms |
| DT1-2 | Wait for 5mins and then call *numSeatsAvailable* API | Load time < 100ms  Size < 100bytes  Response code = 200  Latency < 400ms |

*DT2: findAndHoldSeats API Cases*

*Quick recap: Call to findAndHoldSeats API should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| DT2-1 | A call is made to *findAndHoldSeats* API | Load time < 100ms  Size < 200bytes  Response code = 200  Latency < 400ms |
| DT2-2 | Wait for 5mins and then call *findAndHoldSeats* API | Load time < 100ms  Size < 200bytes  Response code = 200  Latency < 400ms |

*DT3: reserveSeats API Cases*

*Quick recap: Call to reserveSeats API should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| DT3-1 | A call is made to *reserveSeats* API | Load time < 100ms  Size < 100bytes  Response code = 200  Latency < 400ms |
| DT3-2 | Wait for 5mins and then call *reserveSeats* API | Load time < 100ms  Size < 100bytes  Response code = 200  Latency < 400ms |

*DT4: numSeatsAvailable & findAndHoldSeats API Cases*

*Quick recap: Call to numSeatsAvailable API and findAndHoldSeats API in any order should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| DT4-1 | A call is made to *numSeatsAvailable* API and then call *findAndHoldSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT4-2 | Wait for 5mins and then call *numSeatsAvailable* API and then call *findAndHoldSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT4-3 | A call is made to *numSeatsAvailable* API, wait for 5mins and then call *findAndHoldSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT4-4 | Wait for 5mins and then call *numSeatsAvailable* API, wait for 5mins and then call *findAndHoldSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT4-5 | A call is made to *numSeatsAvailable* API and a different thread calls *findAndHoldSeats* API at the same time. | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT4-6 | Wait for 5mins and then call *numSeatsAvailable* API, wait for 5mins and a different thread calls *findAndHoldSeats* API at the same time. | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |

*DT5: findAndHoldSeats & reserveSeats API Cases*

*Quick recap: Call to findAndHoldSeats API and reserveSeats API in any order should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| DT5-1 | A call is made to *findAndHoldSeats* API and then call *reserveSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT5-2 | Wait for 5mins and then call *findAndHoldSeats* API and then call *reserveSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT5-3 | A call is made to *findAndHoldSeats* API, wait for 5mins and then call *reserveSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT5-4 | Wait for 5mins and then call *findAndHoldSeats* API, wait for 5mins and then call *reserveSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT5-5 | A call is made to *findAndHoldSeats* API and a different thread calls *reserveSeats* API at the same time. | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT5-6 | Wait for 5mins and then call *findAndHoldSeats* API, wait for 5mins and a different thread calls *reserveSeats* API at the same time. | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |

*DT6: numSeatsAvailable , findAndHoldSeats and reserveSeats API cases*

*Quick recap: Call to numSeatsAvailable API, findAndHoldSeats API and reserveSeats API in any order should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| DT6-1 | A call is made to *numSeatsAvailable API* and then call *findAndHoldSeats* API and then call *reserveSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT6-2 | Wait for 5mins and then call *numSeatsAvailable* API and then call *findAndHoldSeats* API and then call reserveSeats API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT6-3 | A call is made to *numSeatsAvailable* API , wait for 5mins and then call *findAndHoldSeats* API and then call *reserveSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT6-4 | A call is made to *numSeatsAvailable* API and then call *findAndHoldSeats* API , wait for 5mins and then call *reserveSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT6-5 | Wait for 5mins and then call *numSeatsAvailable* API, wait for 5mins and then call *findAndHoldSeats* API and then call *reserveSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT6-6 | A call is made to *numSeatsAvailable* API, wait for 5mins and then call *findAndHoldSeats* API , wait for 5mins and then call *reserveSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT6-7 | Wait for 5mins and then call *numSeatsAvailable* API , wait for 5mins and then call *findAndHoldSeats* API, wait for 5mins and then call *reserveSeats* API | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |
| DT6-8 | A call is made to *numSeatsAvailable* API, a different thread calls *findAndHoldSeats* API at the same time, a different thread call *reserveSeats* API at the same time | Average Load time < 200ms  Average size < 300bytes  Response code = 200  Average Latency < 400ms |

**Load Test**

**Load test is used to define how the system performs under normal load. Does the system configuration support the normal expected load. Normal load calculation is documented in assumptions. Wait of 5mins is introduced assuming the think time for calling the API will be 5mins. Data for the API call with and without the think time should be the same in ideal case, but will server as a good indicator to calculate average. All tests are executed by a single thread, unless specified otherwise. Tolerance assumed for Load tests will be 5% of the benchmarked values.**

*LT1: numSeatsAvailable API Cases*

*Quick recap: Call to numSeatsAvailable API should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| LT1-1 | 20,000 concurrent calls are made to numSeatsAvailable API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT1-2 | Wait for 5mins and then make 20,000 concurrent calls to *numSeatsAvailable* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |

*LT2: findAndHoldSeats API Cases*

*Quick recap: Call to findAndHoldSeats API should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| LT2-1 | 20,000 concurrent calls are made to *findAndHoldSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT2-2 | Wait for 5mins and then make 20,000 concurrent calls to *findAndHoldSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |

*LT3: reserveSeats API Cases*

*Quick recap: Call to reserveSeats API should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| LT3-1 | 20,000 concurrent calls are made to *reserveSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT3-2 | Wait for 5mins and then make 20,000 concurrent calls to *reserveSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |

*LT4: numSeatsAvailable & findAndHoldSeats API Cases*

*Quick recap: Call to numSeatsAvailable API and findAndHoldSeats API in any order should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| LT4-1 | 20,000 concurrent calls are made to *numSeatsAvailable* API and then to *findAndHoldSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT4-2 | Wait for 5mins and then make 20,000 concurrent calls to *numSeatsAvailable* API and then to *findAndHoldSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT4-3 | 20,000 concurrent calls are made tol *numSeatsAvailable* API, wait for 5mins and then call *findAndHoldSeats API* with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT4-4 | Wait for 5mins and then make 20,000 concurrent calls to *numSeatsAvailable* API, wait for 5mins and then call *findAndHoldSeats API* with a ramp-up time of 5 seconds | Benchmark values +/- 5% tolerance |
| LT4-5 | 10000 concurrent calls are made to *numSeatsAvailable* API and another 10000 concurrent calls are made to *findAndHoldSeats* at the same time with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |

*LT5: findAndHoldSeats & reserveSeats API Cases*

*Quick recap: Call to findAndHoldSeats API and reserveSeats API in any order should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| LT5-1 | 20,000 concurrent calls are made to *findAndHoldSeats* API and then to *reserveSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT5-2 | Wait for 5mins and then make 20,000 concurrent calls to *findAndHoldSeats* API and then to *reserveSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT5-3 | 20,000 concurrent calls are made to *findAndHoldSeats* API, wait for 5mins and then call *reserveSeats API* with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT5-4 | Wait for 5mins and then make 20,000 concurrent calls to *findAndHoldSeats* API, wait for 5mins and then call *reserveSeats API* with a ramp-up time of 5 seconds | Benchmark values +/- 5% tolerance |
| LT5-5 | 10000 concurrent calls are made to *findAndHoldSeats* API and another 10000 concurrent calls are made to *reserveSeats* at the same time with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |

*LT6: numSeatsAvailable , findAndHoldSeats and reserveSeats API cases*

*Quick recap: Call to numSeatsAvailable API, findAndHoldSeats API and reserveSeats API in any order should return the same load on each use case*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| LT6-1 | 20,000 concurrent calls are made to *numSeatsAvailable* API and then to *findAndHoldSeats* API and then to *reserveSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT6-2 | Wait for 5mins and make 20,000 concurrent calls to *numSeatsAvailable* API and then to *findAndHoldSeats* API and then to *reserveSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT6-3 | 20,000 concurrent calls are made to *numSeatsAvailable* API, wait for 5mins and then call *findAndHoldSeats* API and then call *reserveSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT6-4 | 20,000 concurrent calls are made to *numSeatsAvailable* API, and then call *findAndHoldSeats* API, wait for 5mins and then call *reserveSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT6-5 | Wait for 5mins and make 20,000 concurrent calls to *numSeatsAvailable* API, wait for 5mins and then call *findAndHoldSeats* API and then call *reserveSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT6-6 | 20,000 concurrent calls are made to *numSeatsAvailable* API, wait for 5mins and then call *findAndHoldSeats* API, wait for 5mins and then call *reserveSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT6-7 | Wait for 5mins, 20,000 concurrent calls are made to *numSeatsAvailable* API, wait for 5mins and then call *findAndHoldSeats* API, wait for 5mins and then call *reserveSeats* API with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |
| LT6-8 | 6600 concurrent calls are made to *numSeatsAvailable* API, 6600 different calls are made to *findAndHoldSeats* API, 6600 different calls are made to *reserveSeats* API at the same time with a ramp-up time of 1 second | Benchmark values +/- 5% tolerance |

**Stress Test**

**In order to find the crash point of the system stress testing will be performed with the 5 the conditions specified below. Number of calls selected for Stress Test will be equal to maximum load the system can handle + 10% = 22000.**

**Condition 1> Linear ramping during peak load time**

**Condition 2> Linear ramping during low load time**

**Condition 3> Ramp up at 25% load(5500 calls), 50% load(11000) and 75% load(16500) during peak load time**

**Condition 4> Ramp up at 25% load(5500 calls), 50% load(11000) and 75% load(16500) during low load time**

**Condition 5> Repeat Condition 1, Condition 2, Condition 3, Condition 4 from multiple geographical locations**

*ST1: numSeatsAvailable API Cases*

*Quick recap: Stress test by calling only numSeatsAvailable API*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| ST1-1 | 22,000 concurrent calls are made to numSeatsAvailable API with a ramp-up time of 1 second | System will fail and recover after a service restart. Note down the breakpoint. |

*ST2: findAndHoldSeats API Cases*

*Quick recap: Stress test by calling only findAndHoldSeats API*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| ST2-1 | 22,000 concurrent calls are made to *findAndHoldSeats* API with a ramp-up time of 1 second | System will fail and recover after a service restart. Note down the breakpoint. |

*ST3: reserveSeats API Cases*

*Quick recap: Stress test by calling only reserveSeats API*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| ST3-1 | 22,000 concurrent calls are made to *reserveSeats* API with a ramp-up time of 1 second | System will fail and recover after a service restart. Note down the breakpoint. |

*ST4: numSeatsAvailable & findAndHoldSeats API Cases*

*Quick recap: Stress test by calling only numSeatsAvailable API and findAndHoldSeats API*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| ST4-1 | 22,000 concurrent calls are made to *numSeatsAvailable* API and then to *findAndHoldSeats* API with a ramp-up time of 1 second | System will fail and recover after a service restart. Note down the breakpoint. |
| ST4-2 | 11,000 concurrent calls are made to *numSeatsAvailable* API and another 11,000 concurrent calls are made to *findAndHoldSeats* at the same time with a ramp-up time of 1 second | System will fail and recover after a service restart. Note down the breakpoint. |

*ST5: findAndHoldSeats & reserveSeats API Cases*

*Quick recap: Stress test by calling only findAndHoldSeats API and reserveSeats API*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| ST5-1 | 20,000 concurrent calls are made to *findAndHoldSeats* API and then to *reserveSeats* API with a ramp-up time of 1 second | System will fail and recover after a service restart. Note down the breakpoint. |
| ST5-2 | 11,000 concurrent calls are made to *findAndHoldSeats* API and another 11,000 concurrent calls are made to *reserveSeats* at the same time with a ramp-up time of 1 second | System will fail and recover after a service restart. Note down the breakpoint. |

*ST6: numSeatsAvailable, findAndHoldSeats and reserveSeats API cases*

*Quick recap: Stress test by calling all API’s*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| ST6-1 | 20,000 concurrent calls are made to *numSeatsAvailable* API and then to *findAndHoldSeats* API and then to *reserveSeats* API with a ramp-up time of 1 second | System will fail and recover after a service restart. Note down the breakpoint. |
| ST6-2 | 6,600 concurrent calls are made to *numSeatsAvailable* API, 6,600 different calls are made to *findAndHoldSeats* API, 6,600 different calls are made to *reserveSeats* API at the same time with a ramp-up time of 1 second | System will fail and recover after a service restart. Note down the breakpoint. |

**Spike Testing**

**Test with sudden burst of parallel or concurrent burst of calls to check how how the server responds to a sudden spike in the number of calls. Spike testing will be performed with the 5 linear loads and 1 non-linear burst conditions specified below. High Load is assumed as 19,000 calls, Normal load is assumed as 17,000 calls, Low load is assumed to be < 500 calls**

*Linear load*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Beginning | Middle | End |
| Load 1 | High | Normal | Low |
| Load 2 | Low | High | Normal |
| Load 3 | Normal | Low | High |
| Load 4 | Normal | Normal | High |
| Load 5 | Low | Low | High |

*Non-Linear load*

Spike burst at a fixed duration: Spike every 5 mins for 1 hour

Spike burst at a random duration: Spike at random interval for 1 hour

*SP1: numSeatsAvailable API Cases*

*Quick recap: Spike test by calling only numSeatsAvailable API*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| LP1-1 | Concurrent calls are made using linear and non linear loads mentioned above to numSeatsAvailable API with a ramp-up time of 1 second | The server should be able to stabilize and return to normal functioning after the spike passes. |

*SP2: findAndHoldSeats API Cases*

*Quick recap: Spike test by calling only findAndHoldSeats API*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| SP2-1 | Concurrent calls are made using linear and non linear loads mentioned above to *findAndHoldSeats* API with a ramp-up time of 1 second | The server should be able to stabilize and return to normal functioning after the spike passes. |

*SP3: reserveSeats API Cases*

*Quick recap: Spike test by calling only reserveSeats API*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| SP3-1 | Concurrent calls are made using linear and non linear loads mentioned above to *reserveSeats* API with a ramp-up time of 1 second | The server should be able to stabilize and return to normal functioning after the spike passes. |

*SP4: numSeatsAvailable & findAndHoldSeats API Cases*

*Quick recap: Spike test by calling only numSeatsAvailable API and findAndHoldSeats API*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| SP4-1 | Concurrent calls are made using linear and non linear loads mentioned above to *numSeatsAvailable* API and then to *findAndHoldSeats* API with a ramp-up time of 1 second | The server should be able to stabilize and return to normal functioning after the spike passes. |
| SP4-2 | 11,000 concurrent calls are made to *numSeatsAvailable* API and another 11,000 concurrent calls are made to *findAndHoldSeats* at the same time with a ramp-up time of 1 second | The server should be able to stabilize and return to normal functioning after the spike passes. |

*SP5: findAndHoldSeats & reserveSeats API Cases*

*Quick recap: Spike test by calling only findAndHoldSeats API and reserveSeats API*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| SP5-1 | Concurrent calls are made using linear and non linear loads mentioned above to *findAndHoldSeats* API and then to *reserveSeats* API with a ramp-up time of 1 second | The server should be able to stabilize and return to normal functioning after the spike passes. |
| SP5-2 | 11,000 concurrent calls are made to *findAndHoldSeats* API and another 11,000 concurrent calls are made to *reserveSeats* at the same time with a ramp-up time of 1 second | The server should be able to stabilize and return to normal functioning after the spike passes. |

*SP6: numSeatsAvailable, findAndHoldSeats and reserveSeats API cases*

*Quick recap: Spike test by calling all API’s*

|  |  |  |
| --- | --- | --- |
| Test# | Test Case Description | Expected Result |
| SP6-1 | Concurrent calls are made using linear and non linear loads mentioned above to *numSeatsAvailable* API and then to *findAndHoldSeats* API and then to *reserveSeats* API with a ramp-up time of 1 second | The server should be able to stabilize and return to normal functioning after the spike passes. |
| SP6-2 | 6,600 concurrent calls are made to *numSeatsAvailable* API, 6,600 different calls are made to *findAndHoldSeats* API, 6,600 different calls are made to *reserveSeats* API at the same time with a ramp-up time of 1 second | The server should be able to stabilize and return to normal functioning after the spike passes. |

**Endurance Test**

**This test ensures how the system performs over a period of time under load. This helps us identify memory leaks. The scenarios for this test will include scheduling cases under Load and Spike Tests. Assuming show starts on 1/1/2018 and ends on 1/30/2018**

|  |  |  |
| --- | --- | --- |
| Schedule# | Schedule | Expected Result |
| SC1 | Test start date: 60 days before show start, Test end date:30 days before show start, Frequency: everyday at peak hour | Memory leaks should not be observed |
| SC2 | Test start date: 60 days before show start, Test end date:30 days before show start, Frequency: everyday at off hour | Memory leaks should not be observed |
| SC3 | Test start date: 60 days before show start, Test end date:30 days before show start,Frequency: once per week | Memory leaks should not be observed |
| SC4 | Test start date: 60 days before show start, Test end date:30 days before show start,Frequency: every 2 days | Memory leaks should not be observed |
| SC5 | Test start date: 60 days before show start, Test end date:30 days before show start, Frequency: random days | Memory leaks should not be observed |

*Automation:*

Approach 1: JMeter

Each test case can be saved as a JMeter Script and run on the test environment using command line. The results captured will be stored in the database. We can extract the KPI’s from the database and verify them against benchmarks programmatically. We can even load the results in JMeter Listeners to view the overall health of the system.

Approach 2: Cucumber Open Soure

Use Cucumber and convert each use case to test script.Each use case is written in a way to be able to generate Gherkin scripts with little modification and configuration. Analysing and verifying the KPI’s can be done in the similar way as documented in Approach 1.

Summary:

Input to automation scripts should be independent of the tool used. The ultimate goal of the scripts is to capture and run through tests, including all edge cases. Once the results of the tests are captured and stored, we can use any graphing tool to analyse the data and maintain the benchmark.

*To-Do:*

1. Research on better approached for benchmarking
2. Read more case studies on other performance tests that will be useful to avoid system failure
3. Simulate mobile users
4. Document sign off criteria

*Resources:*

1. <https://support.smartbear.com/readyapi/docs/loadui>
2. <https://docs.oracle.com/cd/B31104_02/books/TestGuide/booktitle.html>
3. <https://www.blazemeter.com/blog>
4. https://github.com/cucumber/cucumber/wiki/Background