Dynamic Digital App (DDA)

Performance Testing Approach Document

Initial Draft

Table of Contents

1. Application Summary.

This solution is to build a dynamic, digital new business application (DDA) integrated with the Web Illustrations platform.

Dependencies:

- Web Illustration
- DocuSign API
- CIAM
- Advisor Portal
- Arrow
- AWD

2. Application Performance.

Application must meet or perform below set threshold:

Frontend:

- Page (no call to backend): < 1s
- Page with call to backend: < 8s
- Benchmark against 250 concurrent users.

3. Performance Test Strategy

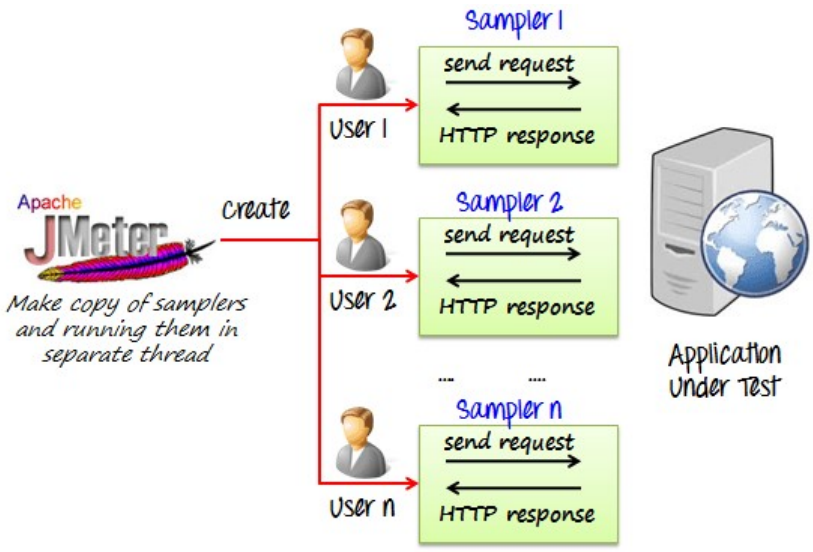
Type of Testing:

Stress test: To determine or validate an application's behavior when it is pushed beyond normal or peak load conditions.

Spike Test: Spike testing is a type of performance testing in which an application receives a sudden and extreme increase or decrease in load. The goal of spike testing is to determine the behavior of a software application when it receives extreme variations in traffic.

Load test: To verify application behavior under normal and peak load conditions.

The figure below shows how JMeter load Testing simulates the heavy load:



Scope

- Load Testing
- Stress Testing
- Spike Testing

Not in Scope

- Endurance Testing

Environmental Requirements

Load Injectors:

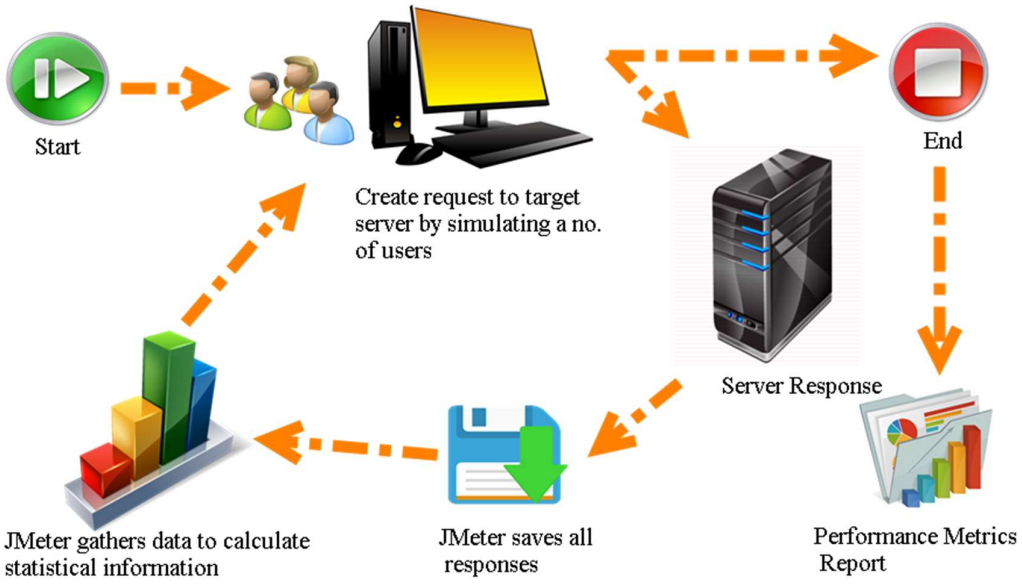
There will be one or more dedicate "load injectors" set up to initiate the required load for performance testing. The load injector could be a VM or multiple VMs that have an instance of JMeter running, initiating the requests.

The performance tests will be run against a stable version of the Policy Service Support Dashboard solution (which has already passed the functional tests) and performed on a dedicated production-like environment (pre-prod?) assigned for performance testing with no deployments on that environment during the course of the performance testing.

Test Tools:

Test tools used for Performance testing will be:

JMeter: An open-source load testing tool. Predominantly used for performance testing.(Load Testing, Stress Testing and Spike Testing).



4. Assumptions Constraints and Risks

Assumptions:

Assumptions should be documented concerning the available release software, test environment, dependencies, tools, and test schedule associated with the performance test. Examples are shown below.

Table 1: Assumptions

No.	Assumption
1	Release x.x Code will be fully functional, having passed Functional and Automated before Performance Testing begins.
2	All Performance Center Tool software has been installed and configured.
3	The fully deployed, installed, and configured Web tier, middleware tier, and database servers must be operational in order for performance testing shake-out to begin.

Constraints:

Constraints should be documented concerning the available release software, test environment, dependencies, tools, test schedule, and other items pertaining to the performance test. Examples are shown below.

Table 2: Constraints

No.	Constraint	Impact
1	The Performance Test environment has 50% of the servers that Production has.	The scaling factor of Performance Test to Production is 50%. All Production Load Models

		that are executed in Performance Test should be run at 50% of the full Production load Model to represent a 100% Load Test in the AJ Test environment.
2	The Performance Test environment does not have some of the older data that Production has, which limits some of the data scenarios that can be simulated.	The data in Production has not been purged; searches in Production intermingle with older data than Performance Test can. This could limit the capability of reproducing some Production issues.
3	The Performance Test team does not have a commercial tool or an approved Wire Shark-like tool that allows for measuring network response times using packet captures.	The impact of network response times will not be measurable as we determine what areas within the Architecture are responsible for transaction response time cost. This constraint will leave network response time cost-related questions unanswered.

Risks:

Risks should be documented concerning the test schedule, release software, dependencies, tools, test approach test environment and other items pertaining to the performance test. Examples are shown below.

Table 3: Risks

No.	Risk	Impact	Action/Mitigation	Assigned To
1	If functional errors from validation testing occur and prevent the creation of performance test scripts or performance test execution, execution of performance test project tasks will be delayed until functional errors can be addressed.	HIGH	The team will start Performance Test execution once environment certification, test script validation, and data staging efforts are completed.	Project Manager
2	If a performance-tuning effort is conducted in the middle of the performance test execution schedule and as a result configuration or code changes are made to the environment, any tests executed prior to the performance-tuning changes should be re-executed.	HIGH	It is recommended that any tests that were executed before the performance tuning changes should be re-executed after the performance-tuning changes.	Project Manager, Performance Engineering

5. Performance Test Data Planning.

Provide a summary of the test data that will be needed for the Performance Test phase. Provide the details of what data will be needed to support the execution of the performance test scripts for several iterations per the Load

Model. There could be needs for dynamic login data and variable data in order to not allow for iterations to be static. The performance test execution iterations should be varied, and the data drives this process.

Data Preparation:

Define the procedure that will be used to prepare the test data for Performance Test. Define the procedures needed to create the test data. Some of the procedures include, but are not limited to:

- Testing the data and database to ensure that they are ready for the current test stage and align with test requirements.
- Identifying/defining any tools needed to create, modify, and manipulate the test data.
- Developing a strategy for obtaining and refreshing test data for every cycle or pass conducted in performance testing. During the performance test, the volume of the test data is large relative to other test stages.

6. Performance Test Monitoring Tools and Metrics.

Tool	Purpose
JMeter	To build performance test scripts To execute performance test scenarios To collect transaction response times

The table below describe examples of the various performance metrics that can be captured during the Performance Test stage to view resource usage trends.

Metrics	Value Measured
CPU utilization	CPU utilization
Physical Memory Percentage used	Physical Memory Percentage used
Memory	Memory utilization
Java Virtual Machine (JVM) Runtime/Total Memory	Total memories in the JVM runtime
JVM Runtime/Free Memory	Free memories in the JVM runtime
	Used memories in the JVM runtime
JDBC Connections/Concurrent Waiters	Number of threads that are currently waiting for connections
JDBC DB Connections/Percent used	Average percent of pool that is in use
JDBC DB Connections/Percent maxed	Average percent of the time that all connections are in use
Thread Creates	Total number of thread creates
Thread Destroys	Total number of threads destroyed
Thread Pool/Active Threads	Number of concurrently active threads

Metrics	Value Measured
Thread Pool/Pool Size	Average number of threads in pool
Thread Pool/Percent Maxed	Average percent of the time that all threads are in use
Heap size	Amount of heap allocated.
Memory	Memory utilization Processes in run queue (Procs r), User Time (cpu us), System time(cpu sy), Idle time (cpu id), Context Switching (cs), Interrupts
Disk I/O	Disk I/O utilization Read/Write per sec (r/s, w/s), Percentage busy (%b), Service Time (svc_t)
Network	Collisions (Collis), Output Packets (Opkts), Input errors (Ierrs), Input Packets (Ipkts)
Queue Depth	Measurement of queue depths during the test execution
Physical Memory Percentage used	Physical Memory Percentage used

DOCUMENT SIGNOFF

Nature of Signoff	Person	Signature	Date	Role
Authors	Navya Tadepalli			Associate Quality Engineer
Reviewers	Wendy Dsouza		05/26/2021	Technical Program Manager

DOCUMENT CHANGE RECORD

Date	Version	Author	Change Details
05/25/2021	Initial Version	Navya Tadepalli	