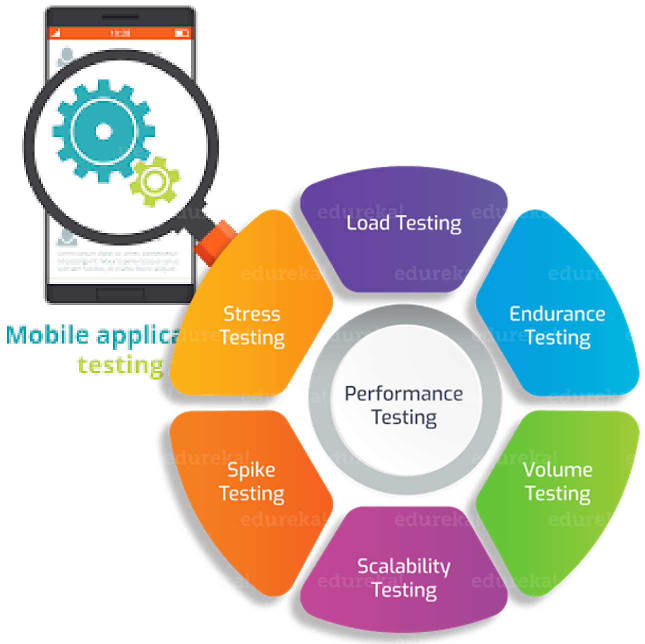


Perf Matrix
Make your app sure

GRAFIKART.FR

Plan de test performance

Version 1.0

28/06/2022

Approbation des tests de performances

Table 1: Sign-off Detail

Name	Role / Designation	Signoff Date	Signature
Jonathan Boyer	Project Manager - Lead		

Enregistrement des modifications

Table 2: Enregistrement des modifications

Version Number	Date	Author/Owner	Description of Change
Draft	28/06/2020	PerfMatrix	Draft version with available details

Table des matières

1. Résumé

Le trafic sur l’application est de taille moyenne, les utilisateurs sont constants et peu nombreux, notre test s'oriente sur de l’endurance testing.

1.1 Aperçu: contexte et portée du projet

Le projet est de proposer un panel de cours et de tutoriels pour la communauté qui suit Grafikart. Le partage gratuit de ses compétences et ses découvertes avec les personnes passionnés par le web est son principal argument. Sur le site on y retrouve donc tous les cours qu’il donne au format écrit ET au format vidéo hébergés sur youtube en fonction des chapitres. Il existe aussi une section commentaires sous chaque chapitre mais aussi une section blog et une section forum en fonction de tous les sujets possibles en web. Le forum est également accessible via Discord dans lequel les utilisateurs peuvent continuer d’échanger et reçoivent les notifications des dernières publications de Grafikart

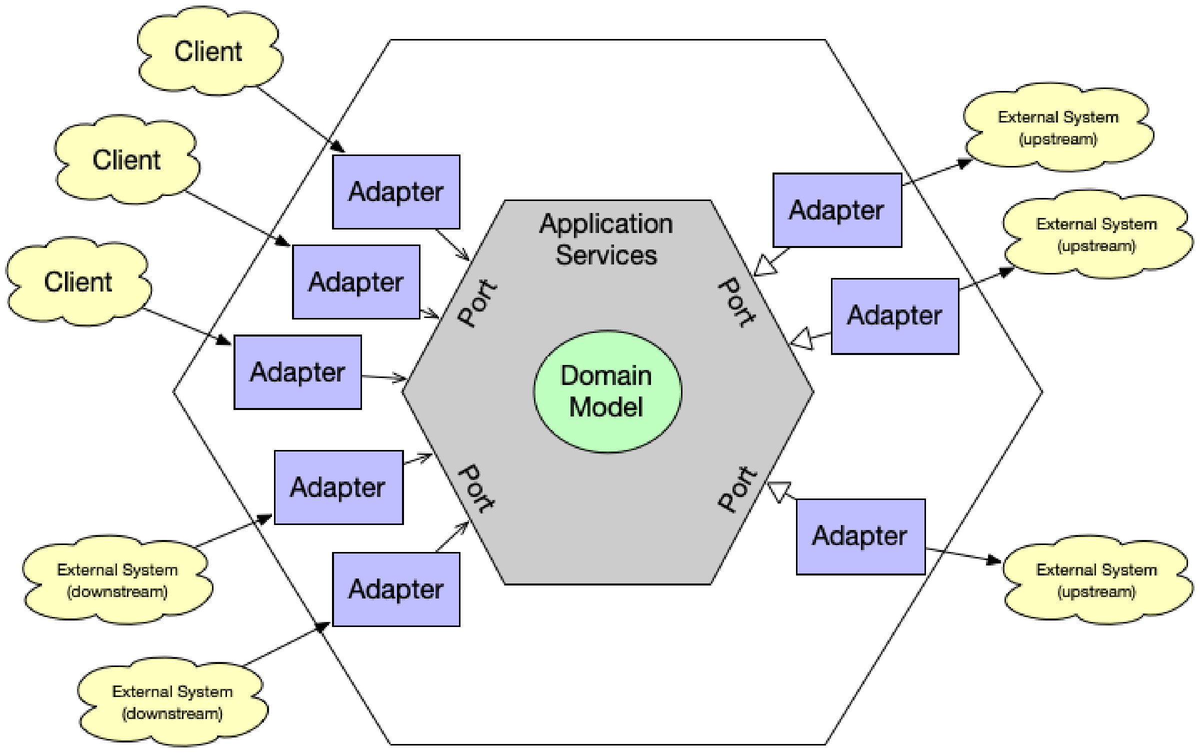
2. Architecture de l’Application

Le backend et la sécurité d’authentification est faite via Symfony, le css est fait maison, et un système de paiement/abonnement pour soutenir le créateur est réalisé avec Stripe

2.1 Overview: System Architecture

- Serveur dédié virtuel avec **Debian**
- **Nginx** pour le serveur web
- **PHP** avec Symfony pour le back-end
- **PostgreSQL** pour la base de donnée
- **Redis** pour le cache, les sessions et le système de fil d’attente
- **Mercure** pour les notifications instantannées
- **Messenger** pour les tâches asynchrones
- **Typesens** pour la recherche
- **NodeJS** pour le bot Discord

2.2 Diagram d'architecture


Client
Client
Client
Adapter
Adapter
Adapter
Adapter
Adapter
Application Services
Domain Model
Port
Port
Port
Port
Adapter
Adapter
Adapter
External System (upstream)
External System (upstream)
External System (upstream)
External System (downstream)
External System (downstream)

2.3 Informations détaillées

Cette structure s'inspire de différents principes tel que le système de contexte, le Domain Driven Development ou encore l'architecture hexagonale sans forcément les respecter à la lettre.

L'idée est de séparer notre application avec 4 dossiers principaux :

- **Domain**, contient les classes qui permettent de gérer la logique métier de l'application. Le domaine doit fonctionner de manière isolée et peut exposer ces méthodes au travers de **Services** ou via les **Repository**.
- **Infrastructure**, définit les éléments d'infrastructure qui permettent au domaine de communiquer avec le système (système de fichiers, envoi d'emails, base de données...).
- **Http**, contient les classes qui permettent d'interagir avec le système depuis des couches HTTP.

- **Command**, contient les commandes qui permettent d'interagir avec le système depuis la CLI.

Dans cette approche, la couche **HTTP** ou **Command** n'est qu'une interface qui permet de communiquer avec notre système.

Comment les choses communiquent ensemble ?

Cette approche fonctionne bien en théorie, mais dans la réalité les choses sont plus complexes et les différents systèmes ont besoin de communiquer les uns avec les autres. Pour cette communication on se repose sur le système d'**évènement** et de **subscriber** du framework.

Lorsque le domaine effectue une opération, un évènement est émis pour pouvoir permettre aux autres systèmes de greffer leur logique.

3. Conditions du test de performance

3.1 Conditions

Nous souhaitons réaliser des tests de performances sur ce site afin de vérifier la viabilité du site en production afin de proposer la meilleure expérience utilisateur possible et par la même occasion détecter d'éventuels goulots d'étranglements pour définir les points faibles de l'application pour définir les zones sur lesquelles des efforts de développement pourront être accrus

3.1.1 Business NFR

Table 3: Business NFR

Business Transactions	**User Load**	**SLA/response times**	**Transactions per hour**
Access landing page	600	2 seconds	3000
Access register page	150	0.5 seconds	800
Access tutorials	300	1 second	1500

4. Performance Test Planning

4.1 Performance Test Approach

<Write a high-level approach for performance testing of the application under test. >

Document the Performance Test Approach by covering five key areas:

1. *You can refer HLD (High-Level Document), LLD (Low-level document), SSD (System Design Document) to collect project background information on the application and infrastructure.*
2. *Understand the Production Load Model to determine what load should also be considered for execution during the performance tests.*
3. *Identify the load test type like Peak hour Load Tests, Stress Tests, Soak Test etc.*
4. *What key performance metric areas will be important to monitor or define the pass criteria? The Performance Requirements are strongly recommended to drive the pass/fail criteria, but previous results comparisons can also be considered in the pass/fail decision process in case of Baseline and Benchmark Testing.*
5. *Consider the change request which was raised in the past. Include the scope of those CRs in this phase of testing. Provide the details of the CR in the below table:*

Table 5: Change Requests (CRs)

Task ID	Description	Project Affected
CRNFT01	Response Time too high for Login Page	XXXXXX
CRNFR02	XXXXX	XXXXXX
CRNFT03	XXXXX	XXXXXX

4.1.1 Performance Testing and Monitoring Tool Details

Table 6: Description of Performance Testing Tool

Tool Name	Description	Licensed / Open-Source?	No. of licenses
Micro Focus Performance Center	Version: 12.55 Required Protocol: Web HTTP/HTML Support Forum Link: Support ID:	Licensed	10,000
Dynatrace	Version 1.1 Support Forum Link: Support ID:	Licensed	NA
xxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxx

4.1.2 Performance Test Script Steps

<In this section, the performance test scripts that need to be developed are detailed by user action step as shown in the tables below. For each key Business Process within the Application under Test which was agreed from the project; a Performance Test script needs to be developed.

The transaction flow and script details must be given like below table: Develop performance test scripts that simulate all of the actions in the Business Processes/Transactions documented in the Load Model.>

Table 7: Performance Test (Script 1 Steps)

Step #	Application Name: PerfMatrix Business Process Name: Product Ordering NFT Script Name: 01_PerfMatrix_ProductOrdering
1	Home Page
2	Login
3	Search Product
4	Select Product
5	Payment
6	Logout

Table 8: Performance Test (Script 2 Steps)

Step #	Application Name: PerfMatrix Business Process Name: Product Cancellation NFT Script Name: 02_PerfMatrix_ProductCancellation
1	Home Page
2	Login
3	Ordered Product
4	Select Product
5	Cancel Product
6	Logout

Table 9: Performance Test Runtime Settings (Optional Information, provide only if available)

Script #	Pacing between Iterations	Think Time between transactions
Script 1	6 seconds (Fixed)	10 seconds (Fixed)
Script 2	5-10 seconds (Random)	5-10 seconds (Random)
Script 3	No Pacing	10 seconds (Fixed)
Script 4	No Pacing	No Think Time (Only 1 transaction in the script)
Script 5	12 seconds (Fixed)	10 seconds (Fixed)
Script 6	12 seconds (Fixed)	10 seconds (Fixed)

4.1.3 Performance Test Data Planning

<Please write a summary of the test data that will be needed during the Performance Test phase. Provide the details of what type of data and how much test data will be needed for all the in-scope business flows. The requirement of test data should include all the cycles as well as releases. Also, need to mention whether test data will be prepared or extracted from the production>

4.1.3.1 Data Preparation

<Please write the procedure that will be used to prepare the test data for Performance Test. Define the procedures needed to create the test data. If there is any third party or external tool required to prepare the test data then name the tool and procedure. If test data is being copied from production then mention the confidentiality of the data.>

5. Performance Test Execution

5.1 Performance Test Summary

<The table below provides an example of a short summary of each of the Performance Test scenario runs.>

Table 10: Performance Test Scenarios

Test Run	Date	Test Scenario Summary
Smoke Test	To Be Determined (TBD)	To validate the performance test scripts and monitors
Cycle 1 - Run 1	TBD	Load Test - 1 Hour test with peak load
Cycle 1 - Run 2	TBD	Repeat Load Test - 1 Hour test with peak load
Cycle 1 - Run 3	TBD	Stress Test - 1 Hour test with 150% of peak load
Cycle 1 - Run 4	TBD	Repeat Stress Test - 1 Hour test with 150% of peak load
Cycle 1 - Run 5	TBD	Soak Test - 8 HourTest with average load
Cycle 1 - Run 6	TBD	Repeat Soak Test - 8 HourTest with average load
Cycle 2 - Run 1	TBD	Load Test - 1 Hour test with peak load
Cycle 2 - Run 2	TBD	Repeat Load Test - 1 Hour test with peak load
Cycle 2 - Run 3	TBD	Stress Test - 1 Hour test with 150% of peak load
Cycle 2 - Run 4	TBD	Repeat Stress Test - 1 Hour test with 150% of peak load
Cycle 2 - Run 5	TBD	Soak Test - 8 HourTest with average load
Cycle 2 - Run 6	TBD	Repeat Soak Test - 8 HourTest with average load

5.2 Performance Test Details

5.2.1 Smoke Test

The smoke test is designed to ensure that the performance test scripts are working in the Performance Test Environment. The smoke test is also used for making sure the Performance Monitors that are configured for metrics collection are operating as expected. The smoke test can also be used to run with 1 to 10 users test to determine how long it takes for transaction steps to complete. This method is valuable for the runtime settings pacing of the test.

5.2.2 Load Test

Table 11: Load Test Scenarios Detail

	Test Details
Test ID	NFT01 (Cycle 1-Run1, Cycle 1-Run2, Cycle 2-Run1 and Cycle 2 Run 1)
Purpose	Peak hour transaction processing will be under examination to determine if the system can maintain response times under the highest anticipated load. This test is designed to collect performance metrics on transaction throughput, response times, and system resource utilization, in comparison to Performance requirements.
No. of Tests	4 (2 tests per cycle)

Duration	Ramp-up: Steady State: Ramp-down:
Scripts	1. XXXXXX 2. XXXXXX 3. XXXXXX
Scenario Name	Load Test Scenario
Covered NFR	NFR01, NFR04 and NFR05
User Load / Volume	500 Vusers (Threads) Load
Entry Criteria	1. The code should be stable and functionally verified 2. Test Environment should be stable and ready to use 3. Test Data should be available 4. All the NFRs should be agreed with the project 5. Test scripts should be ready to use 6. XXXXXX
Exit Criteria	1. All the NFR must be met 2. The error rate of transactions must not be more than 3% of total transaction count 3. CPU utilization must not be more than 60%

5.2.3 Stress Test

Table 12: Stress Test Scenarios Detail

	Test Details
Test ID	NFT02 (Cycle 1-Run 3, Cycle 1-Run 4, Cycle 2-Run 3 and Cycle 2 Run 4)
Purpose	Stressing the system to view if the workload increases in the future then how the application and infrastructure scales. This test will be conducted to determine if response times can be maintained. This test is designed to collect performance metrics on transaction throughput, response times, and system resource utilization, in comparison to Performance requirements.
No. of Tests	4 (2 tests per cycle)
Duration	Ramp-up: Steady State: Ramp-down:
Scripts	1. XXXXXX 2. XXXXXX 3. XXXXXX
Scenario Name	Stress Test Scenario
Covered NFR	NFR02, NFR04 and NFR05
User Load / Volume	750 Vusers (Threads) Load
Entry Criteria	1. The code should be stable and functionally verified

	2. Test Environment should be stable and ready to use 3. Test Data should be available 4. All the NFRs should be agreed with the project 5. Test scripts should be ready to use 6. XXXXXX
Exit Criteria	1. All the NFR must be met 2. The error rate of transactions must not be more than 3% of total transaction count 3. CPU utilization must not be more than 60%

5.2.4 Soak Test

Table 13: Soak Test Scenarios Detail

	Test Details
Test ID	NFT03 (Cycle 1-Run 5, Cycle 1-Run 6, Cycle 2-Run 5 and Cycle 2 Run 6)
Purpose	This soak test will determine if the system resources are recycled for re-use while processing transactions over long periods. Proper recycling of memory, CPU, and other system utilization resources is healthy for performance. This test is designed to collect performance metrics on transaction throughput, response times, and system resource utilization, in comparison to Performance requirements with o memory leakage.
No. of Tests	4 (2 tests per cycle)
Duration	Ramp-up: Steady State: Ramp-down:
Scripts	1. XXXXXX 2. XXXXXX 3. XXXXXX
Scenario Name	Soak Test Scenario
Covered NFR	NFR02, NFR03 and NFR06
User Load / Volume	300 Vusers (Threads) Load
Entry Criteria	1. The code should be stable and functionally verified 2. Test Environment should be stable and ready to use 3. Test Data should be available 4. All the NFRs should be agreed with the project 5. Test scripts should be ready to use 6. XXXXXX
Exit Criteria	1. All the NFR must be met 2. The error rate of transactions must not be more than 3% of total transaction count 3. CPU utilization must not be more than 60% 4. No Memory leakage 5. XXXXXX

5.3 Performance Test Monitoring Metrics

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

Table 14: Application Server Tier

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	The average percentage of the pool that is in use
JDBC DB Connections/Percent maxed	The average percentage 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
Thread Pool/Pool Size	Average number of threads in the pool
Thread Pool/Percent Maxed	The average percentage of the time that all threads are in use
Heap size	Amount of heap allocated.
Memory	Memory utilization Processes in the run queue (Procs r), User Time (CPU US), System time(CPU SV), 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

Table 15: Database Server Tier5

Metrics	Value Measured
CPU utilization	CPU utilization
Physical Memory Percentage used	Physical Memory Percentage used
Memory	Memory utilization Processes in the run queue (Procs r), User Time (CPU US), System time(CPU SV), 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)

5.4 Performance Test Environment

The Performance Test environment is XX% of the production environment. Hence user load has been scaled down to XX%. Post-execution, the test result will be extrapolated with the same percentage.

As listed below, describe what the Scaling factor between the Production environment that will support the Application under Test, and the Performance Test environment that will support the Application under Test.

The Scaling factors are as follows:

1. *Number of CPUs (processors)?*
2. *Memory*
3. *Disk Space*
4. *Load Balancer and its configuration like algorithm*
5. *Environment configuration files – It should be the same in both the environment*
6. *Test Data – It should be populated in the Database to the same level as in Production? If not, what is the ratio?*

Table 16: Performance Test Environment Details

Server Name	Environment Tier	Hardware Version	OS	Memory (GB)	CPU count	Total Disk Space
xxx	Web Service	M620	Linux	32 GB	8 cores	512 GB
xxx	Web Service	M620	Linux	32 GB	8 cores	512 GB
xxx	Middleware	M620	Linux	32 GB	8 cores	512 GB
xxx	Middleware	M620	Linux	32 GB	8 cores	512 GB
xxx	Middleware	M820	Linux	32 GB	16 cores	1 TB
xxx	Database	M820	Linux	32 GB	16 cores	1 TB
xxx	xxx	xxx	xxx	xxx	xxx	xxx
xxx	xxx	xxx	xxx	xxx	xxx	xxx

Table 17: Production Environment Details

Server Name	Environment Tier	Hardware Version	OS	Memory (GB)	CPU count	Total Disk Space
xxx	Web Service	M620	Linux	32 GB	8 cores	512 GB
xxx	Web Service	M620	Linux	32 GB	8 cores	512 GB
xxx	Middleware	M620	Linux	32 GB	8 cores	512 GB
xxx	Middleware	M620	Linux	32 GB	8 cores	512 GB
xxx	Middleware	M820	Linux	32 GB	16 cores	1 TB
xxx	Database	M820	Linux	32 GB	16 cores	1 TB
xxx	xxx	xxx	xxx	xxx	xxx	xxx
xxx	xxx	xxx	xxx	xxx	xxx	xxx

5.5 Assumptions, Constraints, Risks and Dependencies

5.5.1 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 18: Assumptions

No.	Assumption
1	The code version XXXX is stable and passed in functional testing before deploying in the Performance Testing environment.
2	The required license must be available in the Performance Center to run the test.
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.
4	Test Data must be provided to the performance testing team before testing starts
5	xxxxxxxxxx

5.5.2 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 19: Constraints

No.	Constraint	Impact

1	The Performance Test environment has 50% of the servers that Production has.	The scaling factor of the Performance Test to Production is 50%. All Production Load Models that are executed in the 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 since 2000; 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.
4	xxxx	xxxx

5.5.3 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 20: 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
3	xxxx	xxxx	xxxx	xxxx

5.5.4 Dependencies

<Dependencies should be documented concerning the latest build, test data, schedule, required tools' installation, test environment and other items pertaining to the performance test. Examples are shown below.>

Table 21: Risks

No.	Dependencies	Impact	Action/Mitigation	Assigned To
1	The latest build should be available in the non-functional environment before NFT start date	HIGH	The team will start Performance Test execution once the environment has the latest and functionally tested code.	Developer
2	Test data should be provided by the test data team	HIGH	Test data team will fetch the production data and provide to the performance testing team.	Test Data Team
3	xxxx	xxxx	xxxx	xxxx

6. Milestones

Key milestones are listed in the table below. Each of the milestones represents a group of tasks on which completion of Performance Testing is dependent. If any of the milestones are listed as "At Risk", the milestones that follow it will most likely be delayed as well.

Table 22: Schedule of Milestones

ID	% Done	At Risk	Task	Due Date	Interface
1	0-100	Yes or No	Preliminary Project Plan submitted	xx/xx/xxxx	Project Management
2	0-100	Yes or No	Final Project Plan submitted	xx/xx/xxxx	Project Management
3	0-100	Yes or No	Performance Requirements and Production Load Model reviewed and verified	xx/xx/xxxx	Requirements Management and Performance Engineer
4	0-100	Yes or No	Environment Planning	xx/xx/xxxx	Environment Team and Project Management
5	0-100	Yes or No	Test Plan	xx/xx/xxxx	Performance Engineer
6	0-100	Yes or No	Script Development and Data Planning	xx/xx/xxxx	Performance Engineer and Vendor Project Team
7	0-100	Yes or No	Environment Certification and Test Script Validation	xx/xx/xxxx	Project Management and Environment Team
8	0-100	Yes or No	Data Staging and Setup	xx/xx/xxxx	Performance Engineer and Vendor Project Team
9	0-100	Yes or No	Performance Monitoring Configuration	xx/xx/xxxx	Environment Team and Performance Engineer
10	0-100	Yes or No	Test Execution and Analysis	xx/xx/xxxx	Performance Engineer, Monitoring Tool administrators, and Development

6.1.1 Test Organization

<Document the test organization and any other departments that will be supporting the Performance Test Phase.>

Table 23: Test Organization

Name	Functional Role	Responsibilities
Name	Project Manager	Facilitating and coordinating all schedules related to SDLC phases and infrastructure

Name	Performance Engineering Lead	Manages schedules and activities related to Performance Testing projects
Name	Performance Engineer	Prepares for performance test execution, executes performance tests, analyzes performance tests, and tracks problem reports
Name	Performance Engineer	Prepares for performance test execution, executes performance tests, analyzes performance tests, and tracks problem reports.
Name	Monitoring Support	Monitors performance tests using Performance monitors
Name	Application Support	Supports performance test execution as configuration or application issues are found
Name	Performance Test Environment Support	Supports and maintains the Performance Test environment

Appendix A: Acronyms

<List out all the acronyms and associated literal translations used within the document. List the acronyms in alphabetical order using a tabular format as depicted below.

Table 24: Acronyms

Acronym	Literal Translation
NFR	Non-functional Requirement
PT	Performance Testing

Appendix B: Glossary

<Write down the clear and concise definitions for terms used in this document that may be unfamiliar to readers of the document. Terms are to be listed in alphabetical order.>

Table 25: Glossary

Term	Definition
Pacing	The delay between two iterations
Think Time	The delay between two transactions

Appendix C: **Referenced Documents**

<List out the documents which were referred during the preparation of Performance Test plan. Also, provide who and when the reference document was prepared along with version>

Table 26: Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
AO (Architecture Overview) Version: 1.2	https://xxxxxx.xxxxx.com/project_document/architecture/ao.doc	30/10/2018