**Performance Test Strategy**

**Project Name: Project Y**

**Release: R001 Project ID: P002 Enterprise Release ID: 2025.M11**

**Application Name: Release Alpha (MYS)**

**Application ID: APPID-23344**

# Document Change History and Management

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revision History | | | | |
| Revision Number | Author | Revision Date | Description of Changes | Revision Author |
| 1.0 | **Venkat Matta** | **27-March-2025** |  | **Some one** |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[Document Change History and Management 2](#_Toc194137461)

[1. Document Information 5](#_Toc194137462)

[Distribution List 5](#_Toc194137463)

[1 Performance Test Strategy Endorsement 5](#_Toc194137464)

[1.1 Stakeholder Endorsement 5](#_Toc194137465)

[1.2 Recommendation 5](#_Toc194137466)

[1.3 Comments 6](#_Toc194137467)

[Introduction 6](#_Toc194137468)

[1.3.1 Non-Functional Requirements 6](#_Toc194137469)

[1.3.2 Application Architecture Diagram:- 7](#_Toc194137470)

[<Diagram> 7](#_Toc194137471)

[1.3.3 Neon Functional Test 7](#_Toc194137472)

[1.4 Purpose of Performance Test Strategy 7](#_Toc194137473)

[1.5 Intended Audience of the Strategy 7](#_Toc194137474)

[1.6 Glossary 8](#_Toc194137475)

[Performance Testing Process 8](#_Toc194137476)

[1.7 Identify Key Scenarios 9](#_Toc194137477)

[1.8 Identify Work Load 9](#_Toc194137478)

[1.9 Identify Metrics 10](#_Toc194137479)

[1.10 Create Test Cases 10](#_Toc194137480)

[1.11 Simulate Load 11](#_Toc194137481)

[1.11.1 Performance Testing Types 11](#_Toc194137482)

[1.11.2 Performance Test Approach 13](#_Toc194137483)

[13](#_Toc194137484)

[13](#_Toc194137485)

[13](#_Toc194137486)

[13](#_Toc194137487)

[13](#_Toc194137488)

[1.11.3 Performance Test Activates and deliverables 13](#_Toc194137489)

[1.12 Analyze the Results 15](#_Toc194137490)

[Test Environments and Tools 15](#_Toc194137491)

[1.13 Test Environment 15](#_Toc194137492)

[1.14 Testing Tools 15](#_Toc194137493)

[2 Schedule 16](#_Toc194137494)

[2.1 High-level test schedule 16](#_Toc194137495)

[2.2 Schedule dependencies 16](#_Toc194137496)

[Assumptions, dependencies 16](#_Toc194137497)

[Assumptions 16](#_Toc194137498)

[2.3 Dependencies 17](#_Toc194137499)

[Out Of Scope 17](#_Toc194137500)

[Upfront Issues and risks 17](#_Toc194137501)

[2.4 Upfront Issues 17](#_Toc194137502)

[2.5 Risks 17](#_Toc194137503)

# Document Information

|  |  |
| --- | --- |
| Title | <Performance Test Strategy Document Name> |
| Document Purpose | To communicate all the artifacts and elements required to execute performance tests against the **<Project Name>** |
| Version Number | 0.3 |
| Approval State | Initial Draft/Ready for Review |

### Distribution List

|  |  |  |
| --- | --- | --- |
| Name | Area | Action (Info/ Review/ Sign-off) |
| XXXXXX | Project Manager | Review |
| XXXXX | QA Team Manager | Review |
| XXXXXX | Performance Testing team | Review |
| XXXXXX | Project Manager | Review |

1. Performance Test Strategy Endorsement
   1. Stakeholder Endorsement

Your signature below indicates acceptance of the framework and focus of the Performance Test Plan.

* 1. Recommendation

I have reviewed the Performance Test Plan in detail and recommend (tick the appropriate box):

* The Performance Test Plan has been developed in accordance with the user’s criteria and procedures. Testing may proceed as scheduled.
* Approval is given for the Performance Test Plan to proceed, subject to minor alterations as indicated. No further reference to me is required.
* The significant alterations, as indicated, are to be remedied and reviewed by me before commencement of the performance testing.
* Rejection of the entire Performance Test Plan as given.
  1. Comments

Please fill in with your particulars as appropriate:

|  |  |  |
| --- | --- | --- |
| Name | Signature | Date |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction

**<Overview of the Project Name>**

* + 1. Non-Functional Requirements

This application is using the following technologies

< Define all the technologies>

< Description of technologies>

The below one is the architecture diagram



* + 1. Application Architecture Diagram:-

<Diagram>

* + 1. Neon Functional Test

<Diagram>



* 1. Purpose of Performance Test Strategy

This document describes the **<Project Name>** performance testing strategy, along with the specific methodologies and techniques used by the Quality Assurance testing team to plan, organize and manage the testing activities for the Performance testing.

Performance testing poses many challenges and varies a lot from web and client server application testing. Performance Test requires different standards for different purposes and so there is no single uniform testing methodology for Performance Test. The objective here is to simplify the Performance Test testing and come up with a comprehensive and efficient testing strategy.

* 1. Intended Audience of the Strategy

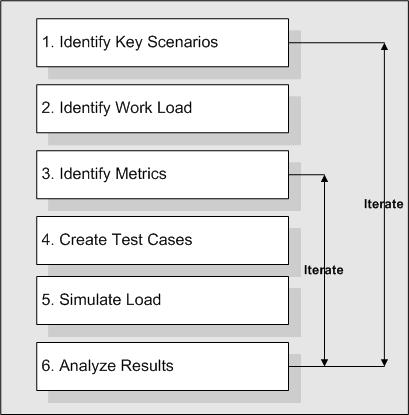
The Intended audiences of the Performance Test Strategy are

* Business Users
* Development Team
* QA Team
  1. Glossary

|  |  |
| --- | --- |
| SLA | Service Level Agreement |
| OLA | Operational Level Agreement |
| Throttling Number | Number of concurrent instances can handle by each web server |
| Test Script | The sequence of functions/code that is generated in Virtual User generator which is used to emulate the real user actions |
| BRD | Business Requirement Document |

# Performance Testing Process

The following steps are involved in performance testing process



* 1. Identify Key Scenarios

It is important to identify application scenarios that are critical for performance rather than testing all the scenarios that are defined by the functional test team. Here the scenarios refer to the web methods that need to be Performance tested. BRD and interaction with Business Analyst will help to identify test cases.

**< Mention all Key scenarios>**

NOTE: - Need to identify more scenarios based on the future requirements

* 1. Identify Work Load

It is important to identify the work loads for the various scenarios mentioned above. In the real world several Performance Test / web methods which will be accessed simultaneously and the load distribution among them are to be identified and simulated in the test process.

The BRD and Functional specifications will be utilized to identify the work load. Any SLA or OLA available for each of the UI Scenario will also drive identifying the work load.

This helps to design the realistic performance test scenario for execution.



< Mention the necessary work load>

Note: - Need to get more performance baselines from client

* 1. Identify Metrics

Identify the metrics that need to be collected during the performance testing. This will help us to come up with the appropriate reports to analyze the test results. The primary metrics that needs to be identified are:

|  |  |
| --- | --- |
| **Throughput** | It is the sum of the data rates that are delivered to all terminals in a network |
| **Response Time** | We need to check the web services and UI response time w.r.to TMS, VSP, **<Project Name>** |
| **Resource utilization** | While doing the Performance testing , we need to check the memory and CPU utilizations of the servers |
| **Maximum concurrent and simultaneous users** | We have to check how many concurrent user can sustain in the environment |
| **System Level metrics** | We need to monitor each individual system resources like Memory, CPU, Disk read/write etc |
| **Network Level metrics** | We need to monitor the network bandwidth w.r.to our SLA’s |

* 1. Create Test Cases

Create comprehensive test cases where the scenarios will be simulated. The test scripts need to be generated for all the test cases and also need to make sure all variants of Performance Test are tested for all the scenarios. The variants which will be tested are:

* UI Scenarios with different security settings (X.509 security / User name & password)
* UI Scenarios /Web service with logging on/off
* UI Scenarios /Web service which has shadow web service calls
* Synchronous / Asynchronous Web service invocation

Test cases will be created for the key scenario’s identified. BRD and interaction with Business Analyst will help to create test cases.

Note: - Need to create the test cases

* 1. Simulate Load

Concurrent user load will be simulated in various patterns using Controller component. Following Load Testing types will be conducted to simulate the **<Project Name>** application load with the production load.

* + 1. Performance Testing Types

|  |  |
| --- | --- |
| Testing Type | Purpose of the testing |
| Load Testing | By doing the load testing we need to verify **<Project Name>** UI, TMS services, VSP server’s behavior under normal and peak load conditions  **Objectives:-** To Check the response times**, throughput rates, resource utilization levels**, |
| Stress Testing | By doing the stress testing we need to evaluate our **<Project Name>** UI, TMS services, VSP server’s behavior when it is pushed beyond the normal or peak load conditions.  **Objectives:-** **synchronization issues and memory leaks**. Stress testing enables you to identify your application's weak points, and how it behaves under extreme load conditions |
| Capacity Testing | By doing the capacity, we can determine the server's ultimate failure point. We can use capacity planning to plan for future growth of the application, such as an increased user base or increased volume of data. For example, to accommodate future loads we need to know how many additional resources (such as CPU, RAM, disk space, or network bandwidth) are necessary to support future usage levels. Capacity testing helps you identify a scaling strategy to determine whether you should scale up or scale out. |
| Web Service Testing | Need to test the each individual web-service and need to find out the system behavior , when we apply the load  **Objective:-** To check the response time and the server utilization |

* + 1. Performance Test Approach

PLANNING PHASE

* Test Plan, Test strategies.
* Selection of testing tools.
* Generating Test Scenarios/Scripts.
* Reviewing the test documents and Base lining.
* Test Readiness Review

REQUIREMENT PHASE

* Requirement Study
* Project Initiation
* Analyzing Test Goals and Objective
* Determining test scope
* Analyzing the H/w and S/W requirements

Design Phase

* Performance Test Scenario
* Preparation Script s and execution
* Collecting the test Data.

ENVIRONMENTAL SETUP

* Set up the environment as per the requirement
* Installation of OS and software and testing tools
* Test readiness review.

EXECUTION PHASE

* Analyze the Data.
* Problem Investigation like bottlenecks (memory, disk, processor, process, cache, network, etc.) resource usage like (memory, CPU, network, etc.,)
* Generate the Performance analysis reports containing all performance attributes of the application.
* Create Recommendation report based on the analysis.
* Repeat the above test for the new build received from client after fixing the bugs

COMPLETION PHASE

* Performance
* Test Report Generation
* Updating test documents
* Post implementation review
* Project completion checklist
  + 1. Performance Test Activates and deliverables

|  |  |
| --- | --- |
| Activities | Responsible Teams |
| Set-up of test environment | Infra Team has to provide the stage environment to do the performance testing. |
| Set-up of test data | Performance team needs to coordinate with manual and dev teams to get the required test data. |
| Performance Test Execution | Performance Team is responsible for doing this testing. |
| Monitoring and collating results | Performance and Infra team need to monitor the servers. Performance team is responsible for gathering the results. |
| Obtaining user acceptance | Need to get customer approval for the performance testing results. |

|  |  |
| --- | --- |
| Deliverables | Responsible Teams |
| Test Strategy | Project Manager, QA Manager and Performance Team |
| Test Plan | QA Manager and Performance TEAM |
| Test Scripts | Performance TEAM |
| Test Executions | Performance TEAM |
| Results | Performance TEAM , Infra Team |

* 1. Analyze the Results

Analyze the metric data captured during the test against the predefined expectations. If the expectations are not met, necessary modifications are made and testing is repeated until the desired results are achieved.



The project is in early stage of development and changes in requirements are expected from the business as well as from within the team. Changes in requirements will force changes in key scenarios and the work loads. Hence the iterative nature of the testing as depicted in the above graphic.

Every test run fine tunes the testing web methods used. As additional metrics are identified there will be additional test cases created.

# Test Environments and Tools

* 1. Test Environment
* Database: We need to specify the database name and versions which we are going to use in the stage environment.
* Web Servers: We need to specify how many web servers and what type of software and hardware, we are going to use in the stage environment.
* App Servers: We need to specify how many app servers and what type of software and hardware, we are going to use in the stage environment.

|  |  |  |
| --- | --- | --- |
| Server Type | Host Name | Description |
| Web Server |  |  |
| App Server |  |  |
| DB Server |  |  |

Note: - Need to get these details from the infra Team regarding the servers hardware and software details

* 1. Testing Tools

The following test tools will be used to achieve the objectives mentioned

|  |  |
| --- | --- |
| **Tools** | **Purpose** |
| Load runner | For creating the Scripts, Test Execution, Analyzing the results |
| Perfmon | Window based Monitoring Tool |
| Site scope | We can configure monitor the entire environment in this tool and it’s a agent less and URL based |
| Quality Center | For writing the test cases ,executions and uploading the scenario documents & results |

NOTE: - Need to finalize about the testing tools

1. Schedule
   1. High-level test schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Test Activity / Milestone | Start Date | End Date | Duration |
| Load Testing |  |  |  |
| Stress Testing |  |  |  |
| Web Service Testing |  |  |  |
| Capacity Testing |  |  |  |

Note: Need to get the testing start and end date

* 1. Schedule dependencies

Document any external dependencies the milestone schedule relies upon,

* Other projects’ delivery schedule
* Previous systems’ release completion
* Third party solution delivery
* Any DB restore’s

# Assumptions, dependencies

## Assumptions

* The test organization should have completed system testing successfully and all high priority errors should have been addressed.
* Email Notification is a static content
* Projects and Targets deletes will be hard deleted from **<Project Name>** UI database
* Data displayed in **<Project Name>** UI will be fetched from **<Project Name>** database for all read operation
  1. Dependencies
* Components to be performance tested shall be completely functional.
* Components to be performance tested shall be housed in hardware/firmware components that are representative or scalable to the intended production systems.
* Data repositories shall be representative or scalable to the intended production systems.
* Performance objectives shall be agreed upon, including working assumptions and testing scenarios.
* Performance testing tools and supporting technologies shall be installed and fully licensed."

# Out Of Scope

Below scenarios are out of scope for performance testing in phase -1 and it may covered in phase -2

|  |  |
| --- | --- |
| **Scenarios** | **Purpose** |

# Upfront Issues and risks

* 1. Upfront Issues

The below issues may occur while doing the performance testing:

* Contention (data, file, memory, processor)
* Inappropriate distribution of workload across available resources
* Inappropriate locking strategy
* Inefficiencies in the application design
* Unexpected increase in transaction rate
* Inefficient use of memory
  1. Risks

|  |  |  |
| --- | --- | --- |
| Risk | Priority | Avoidance |
| Servers not available early during the project | HIGH | We need to use dev. environment to develop single-user scripts |
| Difficulty with Controller licensing, capacity, etc. | MEDIUM | Identify issues early by beginning to use controller as soon as the first small script (such as login only) is coded |
| Not enough capacity in front-end (portal/login) servers. | MEDIUM | Quantify capacity of front-end servers with login only scripts. |
| Servers become unavailable late during the project | LOW | Use production staging environment at night.  Instead of going through load balancer, test directly against one server taken off its cluster |
| Changes in server hardware | LOW | Conduct benchmark tests on hardware as part of project.  Save server configuration files for comparisons |