Document Metadata

| **Project Name** | **Digital Transition Stream** |
| --- | --- |
| **Document Status** | **DRAFT** |
| **Author** | Geo Chirayath |
| **Role** | Performance Test Engineer |
| **Asset(s)** |  |

2.Project Overview

XXXXX

**Objectives:**

* XXX

Document Purpose

Performance Test Strategy document is a high-level document which defines how to carry out performance testing during the testing phase. It tells us how to test a Business requirement and what approach is required to successfully deliver the product to the end client. It also gives a direction that you set at the start of the project with the approach that you are going to take, in order to achieve the Performance testing goals.

The performance test strategy is designed to:

* Act as a control that will contain the agreed scope of activities
* Document activities, resource and the proposed schedule, required to complete the performance testing
* Provide recommendations on the performance test tools that can be used for performance testing
* Advise the requirements and renounces required to support the test effort

Specially, this strategy document includes:

* Tool Recommendation - Test Tools
* How performance test will be tracked - Test schedule
* How it will be tested - Approach
* How the system will be assessed - Based on NFRs
* Where it will be tested - Environment specification
* Who will be involved - Roles and responsibilities

4. Objective

The primary objective of performance testing is to systematically evaluate the responsiveness, stability, scalability and reliability for XX under varying workload models and conditions. This strategy aims to identify and address performance bottlenecks, validate system capacity and ensure optimal user experience by establishing measurable performance benchmark and key performance indicators (KPIs). Through comprehensive testing, we aim to proactively detect and resolve performance issue, ensuring the application meets performance expectations, both in normal operating conditions and during peak usage conditions.

This strategy is designed to:

* **Identify performance bottlenecks**: Identify areas within the application that may impede optimal performance, such as slow response time, resource limitations, or salability constraints.
* **Validate scalability**: Assess the system's ability to scale seamlessly in response to increasing workloads, ensuring it can handle growing user numbers and transaction volumes that are projected.
* **Ensure Stability**: Evaluate the stability of the application under sustained peak load, uncovering any issues related to memory leaks, crashed or unexpected downtime.
* **Establish a Baseline:** Team team will look into running the test in the environment. If this cannot be done, existing results will be compared to the SLA's to derive baseline.
* **Optimize Resource Utilization**: Identify opportunities for resource optimization, ensure efficient use of hardware, software and network resources to enhance overall system, performance.
* **Support Continuous Improvement:**provide

# Strategy of testing

## **5.1 Testing Phases**

* Create **performance test strategy**, to provide an overview of testing activities
* Create **test plan** for each release to cover what is in scope
* Develop **test scripts** for execution
* Source/Prepare **test data** and run **shakeout**, to ensure the scripts work on the intended test environment
* **Run test** to evaluate performance and identify an bottlenecks
* Laisse with project team, to **fix any issues** identified
* Prepare **test summary report**

## **5.2 Acceptance Criteria**

* Non functional requirements are captured and confirmed with stakeholders along with volumetric
* Access to test data and user accounts are provided before scripting
* Application is fully configured and free from any functional defects and is ready for performance testing
* Performance test team has access to application in an isolated environment that is scaled to productions specifications and will be used to run test execution
* Database is primed with production equivalent volumes before test execution
* Test Plan is approved by stakeholders

## **5.3 Completion Criteria**

* All NFRs that are captured as a part of test plan are tested
* Test results are signed off by stakeholders at the end of testing cycle
* Test summary report is generated and signed off by stakeholders
* Source code is maintained in GIT hub for continuous testing and improvements

## **5.4 Reporting**

The following metrics and statistics will be included in the report, these are standard performance trackers and aligns with center of excellence

1. Statistics summary for each transaction:
   * Max response time
   * 95th / 90th percentile response time
   * Avg response time
   * Min response time
2. Transaction summary:
   * Total passed transactions
   * Total failed transactions
3. HTTP response summary
   * Total number of HTTP 2XX responses
   * Total number of HTTP 4XX responses
   * Total number of HTTP 5XX responses
4. Average response time graph
5. 95th / 90th percentile response time graph

Team will also look into implementing real time dashboard using Grafana and Influx DB.

## **5.5 Defect Management**

Defect management is the process of identifying, recording, monitoring and resolving faults.

The goals and objectives of the defect management process is to:

* Provide an effective system for recording defects through JIRA, which arise throughout the testing process and to monitor the progress of their resolution
* Ensure that the defects discovered are promptly prioritised and addressed where applicable
* Provide a mechanism for communication between members for rapid resolution of defects throughout the project
* Provide any feedback on the root cause analysis
* Re-test the defect once rectified

# Testing Scope

## **6.1 In-Scope**

XXX

## **Out of Scope**

XX

# Performance Test Types

The following test types will be executed to measure performance XX infrastructure and are standard and aligns with center of excellence

## **7.1 Peak Load Test**

The objective of this test is to validate the application against projected peak load transactions rates as defined in the NFR document with acceptable transaction response time and resource utilisation. This test will demonstrate the capability of the XX online infrastructure to achieve the performance requirements at the defined peak load.

## **7.2 Stress Test**

The Objective of the test is to access the performance of the application while undergoing stress. Application will be subjected to 2X peak load.

## **7.3 Soak Test**

The objective of this test is to execute low volume (usually 50% of the projected peak load) test over a long period of time (8 hrs). This test aims to identify stability issues in the system, or any deteriorations in application performance over time due to such artefacts such as memory leaks, database management issues etc.

# Environment

TBC

## **8.1 Environment Diagram**

<<TBC>>

## **8.2 Environment Monitoring**

Application monitoring tools will be used to monitor PERF test environment during the test execution to identify where the bottle neck is. Infrastructure / environment team will configure and manage the monitoring tool. Performance testers will be provided with access to the tools during test execution.

# Tools

The tool is yet to be finalised, however the following factors would be considered for its selection.

* Open source: open-source tool that is free to use and can be easily customized and extended by developers.
* Protocol support: Supports a wide range of protocols and technologies, including HTTP, HTTPS, SOAP, REST, FTP, and more.
* User-friendly GUI: Has a user-friendly GUI that allows users to easily create and configure test plans.
* Flexible test execution: Allows users to execute load tests using various load profiles, including constant load, ramp-up, and step load.
* Customizable reporting: Provides customizable reporting options, allowing users to generate reports that meet their needs.
* Large community support: Large and active community of users who contribute to its development and provide resources for support and troubleshooting.

## **9.1 What we have currently**

Current performance test suite is built using Gatling framework. This has been implemented to test both the website and APIs. Gatling is based on Scala programing language and is a very CI/CD friendly framework.

## **9.2 Why are we changing**

|  | **Comments** |
| --- | --- |
| Gatling is a code based tool | Gatling scripting is code based and requires fair idea of functional programming, This can be easier with API testing, however when trying to automate websites this can become challenging, specially with debugging. |
| Experienced skill set | Experienced skill set not readily available in the project |
| Not a standard tools | Not a standard tools used by center of excellence |
| Lack of run time information | There is poor level of information available to identify issues during test execution. |
| Security clearance | Since Gatling is new to XX this will need security clearance |
| Maintainability | It has been noted, maintaining Gatling scripts has been a trouble in the past |
| Language | Need to understand Scala programming language, which would require upskilling |
| Industry Standard Tool | Gatling is not an industry standard tool specially with website testing |
| Free Version | Using the free version of Gatling, the load distribution (running a higher load from a farm of load generator machines) is not so intuitive like other open source tools. |

## **9.3 What are we changing to**

We are exploring two different options at the moment

#### 9.3.1 LoadRunner

| **PROS** | **CONS** |
| --- | --- |
| **PROS** | **CONS** |
| Supports many protocols and technologies, including HTTP, HTTPS, SOAP, REST, and more, making it a versatile tool. This would meet the scripting requirements for digital | License would need to be purchased to run test. Licenses are based on the number of Virtual users required and not based on number of accounts. If licenses are maintained at enterprise level and are available to use, we might not have to purchase specifically for this project. |
| Used by Centre of excellence team for various project. This would enable us to re-use the existing licences (depending on availability) | Can be resource-intensive, and users may require a powerful machine to run tests effectively. Ideally, this is setup at an enterprise level and projects are added to use those resources. This might be already available and ready to use, as it's used by other teams. |
| Provides real-time monitoring of performance metrics, helps in identifying performance issues quickly. LoadRunner comes with real time monitoring graphs and logs available while running the test to better understand the root cause of the problem | May have compatibility issues with operating systems. At the moment scripting in LoadRunnner can only be done on windows machines. |
| Realistic simulation of users: Allows users to simulate many virtual users to create real load scenarios that accurately reflect real-world conditions. LoadRunner comes with wide range of scenario simulation options, which can replicate PROD scenarios more accurately | Has a smaller user community than open-source tools such as JMeter, and users may need support resources for support and troubleshooting. This is usually handled at enterprise level. |
| Experience skillset available. And quick turn around time for scripting. |  |
| LoadRunner comes with inbuilt recorder using browser, which can record scenario much quicker. Other tools would need a proxy/similar to record, which is restricted in NAB. This would mean more manual effort and consumes time |  |

#### 9.3.2 J-Meter

| **PROS** | **CONS** |
| --- | --- |
| Open Source (No cost involved). Also, widely used by Centre of excellence team for various projects. | Scripting turnaround time is as not as quick as LoadRunner. The lack of out of the box recording option, creates the need to manual script creation and correlations.  In-Built recording cannot be used, as it requires proxy (blocked by security) |
| Comes with a large and active community of users who contribute to its development and provide resources for support and troubleshooting. However, many features are supported using plugins and might have security risks. | Limited scripting options: It’s scripting language, BeanShell, has limited options compared to other load-testing tools. |
| Can easily integrate with CI/CD tools like Jenkins for continuous testing. Which makes it a best candidate to integrate with other pipelines jobs. | After a certain limit, high memory consumption causes errors are common. |
| Easy to Install | Less Proficient than paid performance testing tools such as LoadRunner. |
| Has a user-friendly GUI that allows users to easily create and configure test plans, which is much easier compared to other tools like Gatling. | It has a steep learning curve, and users may require significant technical expertise to configure and use the tool effectively. |

## **9.3 Recommendation**

Considering both J-Meter and LoadRunner are both industry standard, the level of security has prevented using any proxy to record scenarios. This would give advantage to loadrunner, which comes with inbuilt recording using browser and can get scripting done much quicker.

Roles And Responsibility

| **Phase** | **Role** | **Responsibility** |
| --- | --- | --- |
| Performance Test Strategy/Plan | Performance Test Engineer | * Requirements Gathering * Write and Deliver Performance Test Strategy/ Plan |
| IT Project Manager | * Assist Performance Tester with access to Project SMEs and information * Coordinate and review Test Strategy |
| Product Owner / BA | * Provide input data details and in-scope scenarios * Provide performance non-functional requirements * Provide detailed technical information on the infrastructure and interfaces to be performance tested * Review Test Strategy |
| Build and Prepare | Performance Test Engineer | * Liaise with SMEs to obtain Business Process navigation and input data for scripting * Identify user accounts and test data required * Develop performance test scripts and test scenarios * Prepare test user accounts and any required test data on PERF environment |
| IT Project Manager | * Assist Performance Tester with access to Project SMEs and information * Act as an escalation point for any Performance Testing issues encountered (eg. test environment issues) |
| Business Analyst | * Provide input related to functionalities of in-scope scenarios |
| Technical Team | * Provide access to the Non Prod for test script development * Coordinate building of performance test environment including any databases * Deliver performance testing environment fully configured, operational, and source systems all integrated |
| Test Execution | Performance Test Engineer | * Execute Performance Tests * Provide status updates to the stakeholders * Raise any defects found * Assist with defect resolution * Report on testing progress |
| IT Project Manager | * Coordinate environment availability with Security Testing team. * Assist Performance Tester with access to Project SMEs and information. * Act as an escalation point for any Performance Testing issues encountered |
| Technical Team | * Provide monitoring access. * Resolve any defects identified during the Performance Test execution * Improve application performance to meet SLAs (if required) * Provide support during test executions. |
| Infrastructure Team | * Informed during test executions. |
| Results Analysis & Reporting | Performance Test Engineer | * Analyse test results * Provide status reports as required * Write and deliver Performance Test Summary Report |

11. Deliverables

| **#** | **Title** | **Responsible Person** | **Frequency** | **Delivery Method** |
| --- | --- | --- | --- | --- |
| 1 | Test Plan | Performance Test Engineer | One time before each testing cycle | Confluence, e-mail |
| 2 | Test Scripting Status | Performance Test Engineer | Weekly | e-mail |
| 3 | Test Execution Report | Performance Test Engineer | After every test / deliveries | e-mail |
| 4 | Bug Report | Performance Test Engineer | After bug detection during test execution | JIRA |
| 5 | Test Summary Report | Performance Test Engineer | After the end of testing cycle | Confluence, e-mail |

12. Performance Testing Schedule

# Considerations

## **13.1 Assumptions**

* + Suitable performance testing tool is finalised before performance testing activity is started
  + Performance test environment is exactly same as the proposed production environment in terms of code and infrastructure
  + Project would provide test data or support in creation of test data for performance testing
  + MFA would be disabled on performance test environment for automation to work.
  + Application monitoring tools would be setup in PERF environment prior to test execution
  + Functional and technical SME's would be available to support during scripting activities
  + Application support team would support performance test team during execution in terms of application troubleshooting, configuration, auto-scaling etc.

## **13.2 Risks**

| **#** | **Risk** | **Severity** |
| --- | --- | --- |
| 1 | Suitable performance test environment is not ready for test execution | High |
| 2 | MFA not disabled could prevent automation (for MFA related functionalities) | High |
| 3 | Any delay in functional readiness | Medium |
| 4 | Monitoring tool not available | Medium |
| 5 | NFRs not finalised | High |
|  |  |  |

## **13.3 Dependencies**

* + NFR need to be defined
  + End to end connected PERF environment to be setup prior to test execution
  + Suitable non prod environment with end to end connectivity is available for scripting
  + Application monitoring is available prior to test execution

# 14. Document Stakeholders

# 15. Document Control