**Spotify**

# **Introduction**

## Description

This document defines testing approach for Spotify process verification. It briefly describes methods and tools to be used to evaluate and tune performance.

## Purpose

The purpose of this document is to outline the approach for Performance Testing team to follow to assure Performance Acceptance Criteria to be defined and met. Specifically, this document details:

* Performance Acceptance Criteria;
* Performance Testing workflow;
* Load Model;
* Test types to be performed;
* Measurements to be collected;
* Tools and infrastructure.

## Scope

This document provides strategy to carry out all performance analysis activities for the project. It briefly describes resources required, including toolset to accomplish test executions, results analysis, and performance tuning. It covers Performance Acceptance Criteria, explains system interaction models to be tested, and describes scripts to be developed. This strategy doesn’t include functional testing, nor does it guarantee any specific performance results. The primary objectives for this testing effort are to:

* Validate that the Performance Acceptance Criteria are met by the system
* Identify and ensure that performance related defects are addressed prior to deployment

### **Systems features and components for testing**

* Web site UI
  + Web UI
* Search functionality:
  + Autosuggest functionality
  + Regular song search
* Login functionality:
  + Sign In
  + Log out

### **Systems features and components which are not tested**

* The functionality of the whole application:
  + Account tariff plan (Family, Student)
  + Payment methods
  + “Car Thing” feature

### **Non-Functional Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Description** | **Values** |
| 1 | Account update | current or expected amount of updates | 1% of 10000 users |
| 2 | Contact Us | current or expected amount of contact us requests | 1% of 10000 users |
| 3 | Project Contact Us | current or expected amount of project contact us requests | 1% of 10000 users |
| 5 | Search | current or expected amount of search requests | 1% of 10000 users |
| 6 | Sign In | current or expected amount of sign ins | 1% of 10000 users |
| 7 | Log out | current or expected amount of log outs | 1% of 10000 users |

## Defining Performance Pass/Fail Criteria

We need to define what constitutes passing the goal and what constitutes falling short of achieving it. The pass/fail criteria should be unambiguous so that they are measurable in absolute terms. Otherwise, the clients may challenge the pass/fail criteria later. The strategy for Spotify is to achieve step by step the goals set.

Step 1: Define user scenarios.

Step 2: Design workloads based on API endpoint profiles.

Step 3: Define Entry/Exit criteria for each workload.

Step 4: Define proper measurement criteria for each workload.

Step 5: Set the pass/fail criteria for each test run with each workload.

Performance parameters and pass/fail criteria may overlap for a set of workloads.

#### **UI (Client-Side)**

|  |  |
| --- | --- |
|  | **Spotify** |
| UI response time for key business actions (by 90 percentile) | * Page load time - static content pages: **~2s** * DOM user interactive time should not exceed: **4s** * DOM Load complete time should not exceed: **3s** * Page load time - dynamic content pages and transactional pages: **4s** |
| Overall page performance | is not lower than 90 |

#### **Search functionality**

|  |  |
| --- | --- |
|  | Spotify |
| Number of concurrent users per hour | 215,277 |
| Number of search actions per hour | 1% of concurrent users number - 2,152 |
| Search response time for key search actions (by 90 percentile) | ~400ms |

#### **Login**

|  |  |
| --- | --- |
|  | **Spotify** |
| Number of concurrent users | 155,000,000 (premium accounts) |
| Number of key business actions per hour | 1% of concurrent users number - 64,583 |
| User actions distribution | Normal |
| System response time (by 90 percentile) | N/A |

## Resource Planning

### **Human Resource**

|  |  |  |
| --- | --- | --- |
| **No.** | **Member** | **Tasks** |
| **1.** | Project Manager | **Manage** the whole project  Define project **directions**  Acquire appropriate resources |
| **2.** | QA | Identifying and describing appropriate test techniques/tools/automation architecture  Verify and assess the Test Approach  **Execute** the tests, **Log** results, **Report** the defects. |
| **3.** | Developer | **Implement**the test cases, test program, test suite etc. |
| **4.** | DevOps | Builds up and ensures and assets are **managed** and **maintained** |

### **System Resource**

|  |  |  |
| --- | --- | --- |
| **No.** | **Resources** | **Descriptions** |
| **1.** | Server | Install the web application under test  This includes a separate web server, database server, and application server if applicable |
| **2.** | Test tool | The testing tool is to automate the testing, simulate the user operation, generate the test results |
| **3.** | Network | Networks include LAN and Internet to simulate the real business and user environment |
| **4.** | Computer | The PC which users often use to connect the web server |

# **Test Environment**

Set up the load-generation tools and the system under test — collectively known as the performance test environment — and ensure that this environment will meet engineering needs. The coordination component of this step typically involves asking managers and administrators to obtain and/or configure equipment and other resources that are not under the direct control of the team or performance tester.

**Value provided:**

* Ensure that the load-generation and performance-test environments are ready when the team needs them.
* Ensure that the entire team knows who to contact for help with performance-testing environment support.
* Ensure that performance testing support staff knows what they are supporting.

**Tasks accomplished:**

* Performance-test environment configured and ready to begin testing.
* Load-generation environment configured and ready to begin testing.
* Support responsibilities assigned.
* Special permissions, time of day for high load tests, etc., determined.

**Coordinate with:**

* System administrators
* Network support
* Database administrators
* Infrastructure support
* Managers of those above
* Development team

# **Schedule & Estimation**

|  |  |  |
| --- | --- | --- |
| **Task** | **Members** | **Estimate effort** |
| **Create the test specification** | Test Designer | 170 man-hour |
| **Perform Test Execution** | Tester, Test Administrator | 80 man-hour |
| **Test Report** | Tester | 10 man-hour |
| **Test Delivery** |  | 20 man-hour |
| **Total** |  | **280 man-hour** |

### **Test Deliverables**

Test deliverables are provided **before** testing phase.

* Test plans document.
* Test cases documents
* Test Design specifications.

Test deliverables are provided **during** the testing

* Test Scripts
* Simulators
* Test Data
* Test Traceability Matrix
* Error logs and execution logs

Test deliverables are provided **after** the testing cycles is over.

* **Test Results/reports**
* Defect Report
* Installation/ Test procedures guidelines
* **Release notes**

# **Performance acceptance criteria**

## Introduction

Acceptance criteria provide the essentials to be satisfied to accept the application by the end user. Performance efforts associated with two sets of acceptance criteria:

* requirements and objectives;
* engagement criteria.

In the sections below, both types are explained in general and in specific detail for the Spotify performance optimization effort. The efforts will be deemed complete when either all performance criteria are met, or any one of the engagement completion criteria is met.

## Identifying the Performance Testing Objectives

Two major goals is set to be addressed by current plan:

1. Assure that future releases do not cause performance degradations and introduce performance improvements

Systems features for testing:

* Case management actions for Free account
* Case management actions for Premium account

## Load model (business scenarios)

This is the general scenario for Spotify. Will be changed/adjusted during performance testing implementation for each system.

Diagram

Description automatically generated

**Schema description:**

In the first testing phase will be checked web application performance with two user account plans: free and premium

Workload is an instrument simulating the real-world environment. It is an integrated part of the test execution process. Once user journeys or API endpoint profiles are known, a detailed workload can be planned. The workload provides in-depth knowledge of behaviors in proposed system. It explains how typical load will spread once the system go live. In addition, the workload helps to understand the requirements in a structured way. Because the system performance capability viewed from different perspectives, like meeting the goals set by the users, impact on the architecture, and growth potential of the system and so on.

# **Workflow**

## Introduction

This section describes the concepts underlying the activities necessary to make performance testing successful within an iterative process, as well as specific, actionable items that can be immediately applied for the Spotify to gain a significant return on this investment. The key to working within an iteration-based work cycle is team coordination. For this reason, performance analysts must be able to adapt what is measured and analyzed per iteration cycle as circumstances change.

**Test case execution:**

**Capacity test:**

**Test case ID:** TC\_Spotify\_WEB\_Free\_Account\_1 - TC\_Spotify\_WEB\_Free\_Account\_9

**Validation test:**

**Test case ID:** TC\_Spotify\_WEB\_Free\_Account\_1 - TC\_Spotify\_WEB\_Free\_Account\_8

**Fixed load test:**

**Test case ID:** TC\_Spotify\_WEB\_Premium\_Duo\_Account\_4 - TC\_Spotify\_WEB\_Premium\_Duo\_Account\_11

# **Performance test types (with priorities)**

## Server-side (API) tests

### Validation test

Priority: Highest

The pipe-clean test is a preparatory task that serves to validate each performance test script in the performance test environment. The test is normally executed for a single use case as a single virtual user for a set period or for a set number of iterations. This execution should ideally be carried out without any other activity on the system to provide a best-case measurement. You can then use the metrics obtained as a baseline to determine the amount of performance degradation that occurs in response to increasing numbers of users and to determine the server and network footprint for each scripted use case.

Baseline tests need to be ran for every iteration.

**Purpose**

 To check that system is ready for performance testing and scripts are OK

### Capacity test

Priority: Highest

Capacity of a system is the highest level of load it can take and handle without:

* Significant response times increase
* Stability decrease

In other words, capacity is the measure of how powerful the system is.

A capacity test complements load testing by determining server’s ultimate failure point. You perform capacity testing in conjunction with capacity planning, which you use to plan for future growth, such as an increased user base or increased volume of data. For example, to accommodate future loads, you need to know how many additional resources (such as processor capacity, memory usage, disk capacity, or network bandwidth) are necessary to support future usage levels. Capacity testing helps to identify a scaling strategy to determine whether system should scale up or scale down.

**Purpose**

To determine how many users and/or transactions a given system will support and still meet performance goals, to find out server capacity, stability under incremental load and scalability of the system. Also, capacity testing results are key points to create performance tests of another types, for example fixed-load or stress.

### Response time (fixed load) test

Priority: High

Fixed load testing is conducted to verify that your application can meet your desired performance objectives; these performance objectives are often specified in a service level agreement (SLA). A load test enables you to measure response times, throughput rates, and resource-utilization levels, to verify application behavior under normal and peak load conditions.

Define the load level vs capacity:

* Low-load (~10% of capacity)
* Mid-load (~45% of capacity)
* High-load (~80% of capacity)
* Or some defined level of load (e.g. production-like, expected, etc.)

Duration should be:

* long enough to make results statistically meaningful
* short enough to avoid biased errors

Benchmark tests should be used as a part of Continuous execution pipeline

**Purpose**

* to get response times (and some other metrics) statistics under different levels of load and compare them against target/previous release (build, sprint, etc.) results
* to check that system under load is stable for a particular period
* (optional) system resources/application profiling for problematic transactions

The goal of checking system stability is correct for any performance test with fixed level of load.