Spotify

Performance Testing Strategy

|  |  |
| --- | --- |
| Author: | Vladyslav Kliucharov |
| Contributors: | Kelis Franks |
| Reviewer | Lorelei Ramsey |
| Reviewer | Harlee Berg |
| Approver | Mykhailo Kurshakov |

## **Introduction**

The purpose of this document is to specifically describe how the performance requirements for the Spotify Website will be tested and verified in a test environment. The document will outline the scenarios, tests, parameters, and data used in evaluating the capacity of the included features. The scope of tests described in the current document is to verify how the website behaves under load stress. All deliverables should be used for internal (in-house).

**Scope of testing**

Spotify website’s server-side of the application will be tested using performance tools. On the client-side, the application will be tested via Web-app.

### Items to be tested

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

### Items not to be tested

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

## **Roles**

This table shows the staffing assumptions for the project.

|  |  |
| --- | --- |
| Human Resources | |
| Workers | Comments |
| **Project Manager:**  Zidan Penn | * Developing a project plan, including activities, milestones, deadlines, resources needed, and cost estimates * Reviewing project proposals to identify potential risks and challenges * Establishing project goals and objectives, and developing a project plan based on input from stakeholders |
| **Developers**  Kirk Sanchez  Raphael Lin  Kasper Warner  Kelis Franks  Lorelei Ramsey  Harlee Berg | * Developing the project scope based on the client’s needs, using concepts such as time, cost, quality, and risk management * Coordinating with client representatives throughout the project to ensure that all parties are in agreement with decisions made along the way * Working with architects, engineers, and contractors to oversee construction projects from start to finish |
| **Performance analytics:**  Ieuan Gregory  Shivani Mcguire | * Building models to test different business scenarios to determine how they would impact the business * Conducting research on new technologies related to their field, such as artificial intelligence or big data analytics * Working with management to develop and implement reporting systems for the company’s operations * Presenting findings from research projects to management teams |
| **QA tester:**  Ieuan Gregory  Shivani Mcguire | * Communicating with management regarding progress on projects or issues that may require their attention * Developing automated testing tools for standard operating procedures to ensure that they are followed consistently over time * Working with programmers to develop new software features and fixing bugs in existing programs * Evaluating the effectiveness of existing quality assurance programs and recommending changes where necessary |
| **Database administrator:**  Lexie Peel  Connah Rollins | * Participating in strategic planning activities to ensure that the database management department’s activities align with company goals * Ensuring that data is backed up regularly to prevent loss due to computer crashes or hardware failure * Designing new databases or modifying existing ones to meet clients’ needs |

## **Test types assumed for conducting**

#### Load test (description)

Load test is a kind of the most regular test to check the benchmark of the application and its components. Usually, is running after finding all critical/major functional bugs and in a stable prod-like environment.

* Purpose:
  + Measure and analyze performance metrics (response time, error rate) and system behavior under defined load.
  + Compare performance metrics with thresholds/previous results.
  + Collected metrics:
    - Measurements, collected during performance test by load tool:
      * Response time statistics (Minimum, average, 90% percentiles, maximum response time, Response time standard deviation).
      * Total, pass and fail counts.
      * System throughput.
      * Throughput/response time trends.

#### Capacity testing (description)

Should be performed to find the number of virtual users which the application support in a stable state. The test can be performed as one of the first main tests and should be performed after significant changes in the application or its configuration.

* Purpose:
  + Measure and analyze performance metrics (response time, error rate) and system behavior under defined load.
  + Compare performance metrics with thresholds/previous results.
  + Collected metrics:
    - Measurements, collected during performance test by load tool:
      * Response time statistics (Minimum, average, 90% percentiles, maximum response time, Response time standard deviation).
      * Total, pass and fail counts.
      * System throughput.
      * Throughput/response time trends.

#### Stress testing (description)

Stress testing is supposed to run occasionally to check the application’s stability under high load. Can be performed close to after the code complete or by special request.

* Purpose:
  + Measure and analyze performance metrics (response time, error rate) and system behavior under defined load.
  + Compare performance metrics with thresholds/previous results.
  + Collected metrics:
    - Measurements, collected during performance test by load tool:
      * Response time statistics (Minimum, average, 90% percentiles, maximum response time, Response time standard deviation).
      * Total, pass and fail counts.
      * System throughput.
      * Throughput/response time trends.

#### Stability testing (description)

Supposed a long time running the test with the load lower than average. Should be performed occasionally after significant code changes or by special request to make sure the application’s responsiveness and key performance indicators do not change significantly after a long time running, and to check on memory leak as well.

* Purpose:
  + Measure and analyze performance metrics (response time, error rate) and system behavior under defined load.
  + Compare performance metrics with thresholds/previous results.
  + Collected metrics:
    - Measurements, collected during performance test by load tool:
      * Response time statistics (Minimum, average, 90% percentiles, maximum response time, Response time standard deviation).
      * Total, pass and fail counts.
      * System throughput.
      * Throughput/response time trends.

## **Environments**

Environment for performance testing:

The number of users supported in the environment, and access roles for each test user.

Software and hardware requirements like operating system, memory, free disk space, number of systems, etc.

The database clone from the prod environment (for example tables, rows, and columns (not use real user data))

Test data backup and restore, including the baseline functionality backup and virtual user's data (for example account credentials and payment methods).

|  |  |
| --- | --- |
| **Name** | **Description** |
| DEV | Used by developers. May have unstable changes and can be used by QAs only in case of some intermediate changes to be checked. |
| QA | Environment for automation testing. Performance, Regression and Smoke suite running. |
| STAGE | Demo and PO verification |
| PROD | Production environment, Security Testing before going Live. |

## **Tools**

For performance testing, platform usage is supposed.

It includes:

|  |  |  |
| --- | --- | --- |
| **Module** | **Software** | **Description** |
| Metrics visualization | Grafana | The module that visualizes all metrics on the custom dashboards |
| Load generator | Apache Jmeter (NON-GUI) | Module for load-scripts execution., API testing |
| Script creation tool | Apache Jmeter | Module for load-scripts creation., API testing. |

# **Risk Analysis**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk ID | Risks description | Probability | Influence | Effect on Cost/ Schedule/Quality |
| 1 | Late submission of information, delays in document approval by the Customer | Medium | High | Schedule |
| 2 | Incorrect or incomplete stated requirements | High | High | Cost, Schedule |
| 3 | Additional changes in the requirements during development | High | High | Cost, Schedule |
| 4 | Tight time limits that influence the testing flow | High | High | Cost, Schedule, Quality |
| 5 | QA Lead / developer fell ill at the most critical time. No time for staff training. | Low | High | Schedule |
| 6 | The number of bugs significantly exceeds the expected number. | High | High | Cost, Schedule, Quality |

Reduce the level of possible risks:

1) We Need to clarify the person from the customer side who will have the power to resolve emergency cases

2) The requirements should be verified before development has started

3) To resolve additional changes in the requirements during development it should be discussed with the dev team and architect team, how it will influence on the system in the future

4) To resolve the tight time limits, testing should be started as faster as can, and test documentation should be created in the early stage, in case a lack of time only on crucial features should be tested

5) In case QA Lead/developer fell ill, part of the crucial work should be assigned to another team member with the appropriate ability to resolve case, if the case has not high priority can be put on hold

6) In case of a number of bugs significantly exceeds the expected quantity, the dev team should rest assured that is not a server-side issue or DB. QA team should make sanity testing. the last added features. A possible solution restores the system to the previous version

# **Requirements**

In a tables below basic requirements for 2 ways of performance testing (API) will be specified.

#### Common

|  |  |  |
| --- | --- | --- |
|  | **Free account** | **Premium account** |
| Acceptable failure rate | ~2% | ~2% |

Key business actions - actions, which is common for current and target system should be defined here.

#### API (Backend, Server-Side) - testing of application’s APIs under specified level of load.

|  |  |  |
| --- | --- | --- |
|  | **Free account** | **Premium account** |
| Number of concurrent users | 100 | 100 |
| API response time\* for key business actions (by 90 percentile) | 2 sec | 2 sec |

API response time - server response time (time from moment where api request was sent by a client till last response byte was received).