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) usage only and should not be presented to the public.

## **Objectives**

* Verify that future releases will not cause performance degradation, introducing of performance testing improvements
* Measure and analyze end user experience: how quickly application performs from end user standpoint.
* Measure and analyze system behavior under high 80% of capacity production-like load.

## **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 UI.

### Items to be tested

* Application-level
* Client-side performance

### Items not to be tested

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

## **Test Levels**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Description** | **Performed by** | **Environment** |
| **Unit** | Is a method by which individual units of source code are tested to determine if they are fit for use. | DEV | CI/CD |
| **System Functional Testing** | Verifies that functionality meets specified requirements. This testing will determine if the results generated by information systems and their components are accurate and that the system performs according to specifications. | QA | SIT |
| **API Testing** | Testing application programming interfaces directly to determine if they meet expectations for functionality, reliability, performance. | QA | SIT |
| **Accessibility Testing** | Ensure application is accessibility with Disabilities Act. | QA | SIT |
| **Automation Testing** | Verifies that functionality meets specified requirements using automation tools. | QA | SIT for automation |
| **Regression** | Ensure addition of new functionality and defect fixes do not break the existing quality confirmed use-cases. | QA | SIT |
| **Security** | Identify any security risk that the system may have. | Security Services | Prod like environment |
| **Performance** | Ensure system response is within acceptable parameter as per business requirement for single user. | QA | Prod like environment |

## **Roles**

This table shows the staffing assumptions for the project.

|  |  |
| --- | --- |
| Human Resources | |
| Workers | Comments |
| **Project Manager:**  Zidan Penn |  |
| **Developer (Front-end):**  Kirk Sanchez  Raphael Lin  Kasper Warner |  |
| **Developer (Back-end):**  Kelis Franks  Lorelei Ramsey  Harlee Berg |  |
| **Performance analytics:**  Ieuan Gregory  Shivani Mcguire |  |
| **QA tester:**  Ieuan Gregory  Shivani Mcguire |  |
| **Database administrator:**  Lexie Peel  Connah Rollins |  |

## **Test types assumed for conducting**

### API (Backend) performance tests:

#### Validation test (mandatory)

* Purpose:
  + Verify that load script works correctly.
  + Verify than infrastructure for performance tests is prepared correctly
  + Verify that performance tests works as expected, results are reachable in the expected way.
* Load profile:
  + Duration: TBD.
* Entry criteria:
  + Testing environment is prepared.
  + Valid code version is deployed to testing environment.
  + Load test scripts are ready.
  + Load generation infrastructure is prepared.
* Success criteria:
  + This scenario will be successfully completed when:
    - Performance tests work as expected.

#### Load test (mandatory)

* Purpose:
  + Measure and analyze performance metrics (response time, error rate) and system behavior under defined load.
  + Compare performance metrics with thresholds/previous results.
  + Load profile:
    - Workload: 75 users .
    - Duration: usually: 15 min.
    - Target throughput: 15 req/s.
  + 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.
  + Entry criteria:
    - Testing environment is prepared.
    - Valid code version is deployed to testing environment.
    - Load test scripts are ready.
    - Load generation infrastructure is prepared.
  + Success criteria:
    - This scenario will be successfully completed when:
      * No unplanned system crashes or exceptions during load test run.
      * Response time metrics meet desired threshold.
      * All observed failures are investigated and understood.

#### 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.

#### Capacity testing (mandatory)

* Purpose:
  + Measure and analyze performance metrics (response time, error rate) and system behavior under defined load.
  + Compare performance metrics with thresholds/previous results.
  + Load profile:
    - Workload: 100 users.
    - Duration: usually: 15 min.
    - Target throughput: 15 req/s.
  + 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.
  + Entry criteria:
    - Testing environment is prepared.
    - Valid code version is deployed to testing environment.
    - Load test scripts are ready.
    - Load generation infrastructure is prepared.
  + Success criteria:
    - This scenario will be successfully completed when:
      * No unplanned system crashes or exceptions during load test run.
      * Response time metrics meet desired threshold.
      * All observed failures are investigated and understood.

#### 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.

#### Stress testing (mandatory)

* Purpose:
  + Measure and analyze performance metrics (response time, error rate) and system behavior under defined load.
  + Compare performance metrics with thresholds/previous results.
  + Load profile:
    - Workload: 150 users.
    - Duration: usually: 15 min.
    - Target throughput: 15 req/s.
  + 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.
  + Entry criteria:
    - Testing environment is prepared.
    - Valid code version is deployed to testing environment.
    - Load test scripts are ready.
    - Load generation infrastructure is prepared.
  + Success criteria:
    - This scenario will be successfully completed when:
      * No unplanned system crashes or exceptions during load test run.
      * Response time metrics meet desired threshold.
      * All observed failures are investigated and understood.

#### 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.

#### Durable (Stability) testing (mandatory)

* Purpose:
  + Measure and analyze performance metrics (response time, error rate) and system behavior under defined load.
  + Compare performance metrics with thresholds/previous results.
  + Load profile:
    - Workload: 55 users.
    - Duration: usually: 10 h.
    - Target throughput: 20 req/s.
  + 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.
  + Entry criteria:
    - Testing environment is prepared.
    - Valid code version is deployed to testing environment.
    - Load test scripts are ready.
    - Load generation infrastructure is prepared.
  + Success criteria:
    - This scenario will be successfully completed when:
      * No unplanned system crashes or exceptions during load test run.
      * Response time metrics meet desired threshold.
      * All observed failures are investigated and understood.

#### Durable (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.

## **Environments**

Environment for performance testing – prod-like environment with migrated content and user data.

## **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) |  |
| Script creation tool | Apache Jmeter | Module for load-scripts creation., API testing. |
| Client-side performance tool | Sitespeed.io | Tools for checking client-side performance |
| Client-side performance tool | Lighthouse | Lighthouse tool is used for additional client-side performance testing with chrome browser |

# **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 restore 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% |
| Level of acceptable degradation | 10% | 10% |

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).