154a, Borschagivska str., Kiev, Ukraine
ph: +38(044)501-55-48
www.qatestlab.com
contact@qa-testlab.com

QATestLab
quality is a rule

Performance Test Plan

General information	
Customer	
Developer	
Created by (Author)	
Preparation date	18.07.2016
Version	1.0
Status	For Approval

Revision History					
Version	**Description**	**Author**	**Date**	**Approved by**	
				Author	**Date**
0.1	Document creation.		19.05.2016		
1.0	Update requirements for the testing.		12.07.2016		

154a, Borschagivska str., Kiev, Ukraine
ph: +38(044)501-55-48
www.qatestlab.com
contact@qa-testlab.com

QATestLab
quality is a rule

Content

1. Introduction

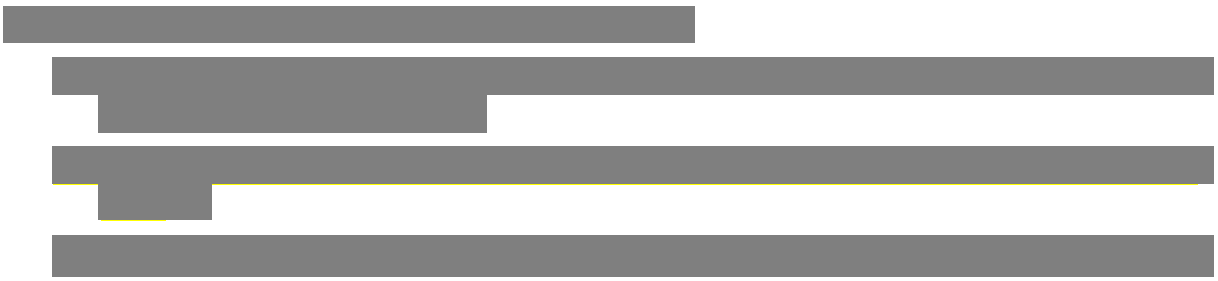
1.1. Purpose of the document

This document describes a test plan for the project "[redacted]" and approaches, which the test team will use to verify the quality of the product. The document also lists the different resources that are needed for a successful performance testing of the project.

1.2. Objective

The purpose of the test plan is to formalize the testing process, plans and approaches to testing, interfacing process with the development team and the project team to achieve the high quality of the software product. The plan takes into account the specifics of the functionality of the project "[redacted]"

2. [redacted]



3. Scope of work

3.1. The components and functions to be tested

№	Components/Applications name	Functions	Link
1	*Front end*	*- Website navigation.*	

3.2. The components and functions not to be tested

№	Components/Applications name	Functions	Comment
1	*Back end*		*Purpose of performance testing is testing web application under load generated by the certain amount of users on the front end.*
2	*Connected 3rd party services*	*Services intended for metrics collection, performance monitoring and infrastructure maintenance.*	*These services are connected for additional needs and are not related to the performance test scenario.*

4. Quality criteria

The delivered product must work in accordance with the requirements and the functional specification listed in sections “Scope of Work “.

The delivered product must not contain any known defects with critical and high priority in the final version.

5. The decisive factors of the project success

- The application should not include known defects with critical and high priority at the time of the final version.
- The application correctly handles required amount of load, without any errors or performance issues.

6. Limitations, assumptions and risks

- The late submission of information or delays in document approval by the Customer.
- Changes in the requirements for performance testing during the testing process.
- Ambiguous requirements can increase the risk of insufficient coverage of functionality by performance testing or risks when input test data or test scenario does not reflect actual product requirements or usage.
- The narrow time frame increases the risk of bugs appearance during performance script development and testing. If the timing of development and environment preparation phases are not met, it will directly affect the timing of testing.
- Unformed or not formed enough team from the Customer's side, which is responsible for monitoring the infrastructure of the application during performance testing, may lead to incorrect performance testing results and application breakdown.

6.1. The risks of the project

Risk Id	Risk description	Probability (High / Medium /Low)	Influence (High / Average / Low)	Effects on Cost / Schedule / Quality
RI.1	The late submission of information, delays in document approval by the Customer.	Medium	High	Schedule
RI.2	Incorrect or incomplete stated requirements for testing.	High	High	Cost, Schedule
RI.3	Changes in the requirements during testing.	High	High	Cost, Schedule
RI.4	Problems with application infrastructure configuration, unavailability of servers.	High	Medium	Schedule
RI.5	Errors in the 3d party performance monitoring tools of the software.	Low	High	Schedule, Quality
RI.6	The narrow time frames. If the timing of development and environment preparation phases are not met, it will directly affect the timing of testing.	Medium	High	Cost, Schedule, Quality
RI.7	Unformed or not formed enough team from the Customer side, which is responsible for monitoring the infrastructure of the application during performance testing.	Medium	High	Schedule, Quality

Risk Id	Risk description	Probability (High / Medium /Low)	Influence (High / Average / Low)	Effects on Cost / Schedule / Quality
RI.8	Insufficient amount or incorrect amount of statistics from the application that may lead to incorrect or incomplete performance testing.	Medium	High	Quality

6.2. Plan to reduce the risks

Risk Id	Actions to reduce the risk
RI.1	Compliance with the rules of planning and organizing meetings. Timely information about the unavailability of employees (including due to vacation, illness, etc.). The schedule of meetings and the provision of necessary information in advance.
RI.2	Splitting testing into several iterations. Frequent testing results discussions.
RI.3	Fixing the basic list of requirements in the contract.
RI.4	Getting further details on installing the product from the Customer's IT department as soon as possible.
RI.5	The provision of an initial stage of development for defining and studying architecture.
RI.6	Follow the development schedule. Timely notification of potential problems or shifts in the schedule.
RI.7	Pre form a team of developers, system administrators and testers from the Customer's side before performance testing.
RI.8	Provide detailed statistics from application and infrastructure about application usage for a long period of time (a few months at least).

6.3. Assumptions

All requirements for performance testing are not yet defined in detail. Estimates made on the basis of how the QATestLab sees the system at the time of the analysis requirements. Estimates may change (increase or decrease) depending on the appearance of new requirements for the system.

7. Resources

7.1. The team of external testing

Company	Name	Role	Contact Information
QATestLab			
QATestLab			

7.2. Tools and services for testing

№	Tool	Comment
1	New Relic	Performance management system for statistics collection from application under test.
2	Apache JMeter	Performance testing tool for performance scripts development and execution.
3	Amazon EC2	Cloud hosting service for load infrastructure setup.

8. Deliverables

8.1. Testing Documentation and Reports

№	Title	Responsible person	Frequency (delivery time)	Delivery method
1	Test Plan	QA Lead	One time before testing	e-mail
2	Scenario performance testing	QA Lead	Upon receipt of the final version of specification	e-mail
3	Bug reports	QA Lead QA Team	After bug detection	e-mail
4	Reports on the results of testing	QA Lead QA Team	After every test / deliveries	e-mail
5	Source code of testing scripts	QA Lead	After all tests	e-mail

9. Strategy of testing

9.1. Testing phases

Main stages of work of the testing team:

1. The testing team gets information about the application (access to the application, testing data) and check what can be tested in case of performance testing.
2. Collect initial statistics information from the application that can be used for performance test plan preparation and performance scenario development.
3. Prepare performance test scenario and confirm it with the Client. Make time estimates needed for testing script development and give the approximate time needed to perform these tests for the desired amount of virtual users.
4. Record and correct testing scripts.
5. Execute the script using low amount of virtual users and generate sample report.
6. Update performance testing scripts if needed.
7. Find the suitable time to provide the main part of the testing. The Client organizes a team from his side: system administrator, programmers and testers, everyone who will monitor the health status of the application and servers and can tweak or reboot infrastructure in case of any problems.
8. The testing team prepares load infrastructure before the testing depending on the statistics of usage from step 2.
9. Run testing script using specified amount of virtual users according to actual statistical information from the application using information from Test iterations section. The Client's team is monitoring the application.

10. After the testing is done, generate execution report and send it to the Client.
11. Repeat steps 6-10 to conduct stress testing of the application using information from Test iterations section.

9.2. Acceptance criteria

1. Requirements for performance testing are received and confirmed.
2. Testing team has access to the application, has all required test data (test accounts, input data).
3. The system is fully configured and ready for performance testing. In the case of "development" or "testing" environment, it is configured in the same way as "product" environment.
4. Test data is loaded into the database of the application in the amount enough for performance testing.
5. The client assigns the task to the testing team.

Test team can partially or completely suspend work, if the following occurs:

1. There is an error in functionality, which does not allow continuing testing.
2. There is a serious problem that prevents the continuation of testing (non-working or damaged test environment, force majeure, such as turning off the Internet or electricity).
3. The developers have not corrected the problem that blocked the testing.

9.3. Completion criteria

1. All test scenarios of the plan for performance testing were performed, performance testing is conducted.
2. Performance testing reports are prepared and sent to the Client.
3. The source code of performance scripts is sent to the Client.

9.4. Testing Methods

Performance testing is regarded as the primary method of testing the application.

9.5. Test Types

9.5.1. Performance testing.

The goal of performance testing to verify the stability of ABC part of the project under expected load following approved test scenarios.

9.6. Reporting

The tools described in Tools and services for testing section will be used to collect the results. Metrics and statistics will be included in the reports, including:

1. Statistics summary:
   - Maximum running concurrent users
   - Total throughput
   - Average throughput
   - Average hits per second
   - HTTP responses summary
2. Transactions summary:
   - Total passed transactions

- Total failed transactions

3. HTTP responses summary:
   - Total amount of HTTP 2XX responses
   - Total amount of HTTP 4XX responses
   - Total amount of HTTP 5XX responses
4. Running concurrent users graph
5. Response times graph
6. Server resources utilization summary:
   - CPU utilization graph
   - Physical memory utilization graph
   - Disk I/O utilization graph
   - Network I/O (Kb/s) graph
   - Top 5 memory consumers graph
   - Top 5 CPU consumers graph

The reports contain metrics and statistics described above, a list of issues (with description and links to statistics section) that occurred during tests execution, possible solutions to increase stability and performance of the application if needed, general conclusion about the performance of the application.

The reports are prepared by the testing team after each iteration of performance scripts execution and sent to the Customer.

10. Requirements for the application for performance testing

The following requirements for the application and load amount values for the testing are under consideration and may change later. These data will be updated according to the usage statistics of the application.

The application must meet the following requirements:

1. The application must respond without errors.
2. The application is required to be available 24 hours per day every day.
3. All user transactions must respond to the user within 3 seconds.

10.1. Requirements calculations

Requirements for the application, performance test scenario and test iterations are calculated according to the statistics taken from performance management system described in Tools and services for testing section.

Required statistical values are taken from New Relic dashboard using NRQL queries described in this section.

10.1.1. Expected transactions time

Expected transactions time value is used to determine the time for one server request.

Transaction average time (TAG):

```
SELECT average(duration) FROM Transaction WHERE appName='OLS-WS-API-
NEW PROD' SINCE 2 months AGO
```

Page view average time (VAG):

```
SELECT average(duration) FROM PageView WHERE appName LIKE 'OLS PROD%'
SINCE 2 months AGO
```

Expected transactions time = TAG + VAG.

Value calculation:

TAG = 0.64 s

VAG = 2.12 s

Average transaction time = 0.64 s + 2.12 s = 2.76 s ~ **3 seconds**

10.1.2. User sessions amount in average

User sessions amount values are used to determine the amount of concurrent users that need to be generated by performance scripts.

User sessions amount per day (for 2 months):

```
SELECT uniqueCount(session) FROM  PageView SINCE 2 months AGO FACET
dayOfMonthOf(timestamp)
```

Value calculation:

User sessions amount per day, in maximum (for 2 month) = 130 000 (sessions/2 months)

User sessions amount per day, in maximum = 130 000 (sessions/2 months) / 2 month = 65 000 sessions

User sessions amount per hour, in average = 74 000 sessions / 24 hours = **2 700 sessions**

10.1.3. Amount of page views per user in average

The value is used to determine amount of requests that must be generated by performance scripts for each concurrent user.

Page views amount (PV):

```
SELECT count(*) FROM  PageView SINCE 2 months AGO
```

User sessions amount (US):

```
SELECT uniqueCount(session) FROM PageView SINCE 2 months AGO
```

Page views per user = PV / US.

Value calculation:

PV = 9.78 M views

US = 2.45 M sessions

Page views per user = 9.78 M views / 2.45 M sessions = 3.9 ~ **4 views per session**

10.1.4. Expected script execution time and amount of concurrent users

Values of script execution time and amount of concurrent users are used for proper load testing configuration. The values are used by load generation tools described in Tools and services for testing section.

Script execution time (iteration execution time) – amount of time needed to execute performance script once by one user:

Script execution time = Average transaction time * Page views per user.

Amount of script sessions per hour – the value that describes how many times the script can be executed by one concurrent user (how many sessions can be generated in one hour):

Amount of script sessions per hour = 1 hour / Script execution time.

Amount of concurrent threads – how many parallel threads need to be prepared to run the same performance script concurrently to generate required amount of sessions in one hour:

Amount of concurrent threads = User sessions amount per hour / Amount of script iterations per hour

Values calculation:

Script execution time = 3 seconds * 4 views per session = **12 seconds**

Amount of script sessions per hour = 3 600 seconds / 12 seconds = **300 (sessions per hour / thread)**

Amount of concurrent threads = 2 700 sessions per hour / 300 (sessions per hour / thread) = 9 **concurrent threads**

10.1.5. Most requested resources of the application

The value is used to determine a list of requests that must be simulated during performance testing:

```
SELECT count(*) FROM PageView SINCE 2 months AGO FACET pageUrl
```

10.1.6. Periods of peak loads

These values are used to determine maximum expected load for the application.

Load amount by hour:

```
SELECT count(*) FROM Transaction SINCE 2 months AGO FACET
hourOf(timestamp)
```

Load amount by day of week:

```
SELECT count(*) FROM Transaction SINCE 2 months AGO FACET
weekdayOf(timestamp)
```

10.1.7. Server resources utilization statistics

This data is needed to determine what amount of load should be generated for stress testing.

10.2. Test iterations

10.2.1. Sample test run

This iteration is needed to check performance testing script before producing a large amount of load to the application. If it is found that prepared performance script doesn't meet approved performance test scenario, then it must be corrected and iteration conducted again.

№	Operation description	Time (minutes)
1	Initialize first 5 concurrent threads.	1
2	Increase load by 1 concurrent thread per minute.	5
3	Continue test execution using 10 concurrent threads.	10
4	Finish test execution, gradually stop concurrent threads.	10

10.2.2. Main test run

This iteration is needed to check application stability using actual amount of load. If it is found that prepared performance script doesn't meet approved performance test scenario, then it must be corrected and iteration conducted again.

№	Operation description	Time (minutes)
1	Initialize first 10 concurrent threads.	1
2	Increase load by 1 concurrent thread per minute.	10
3	Continue test execution using 20 concurrent threads.	60
4	Finish test execution, gradually stop concurrent threads.	10

10.2.3. Stress test run

This iteration is needed to check application stability using the amount of load above actual values. In the case of application instability or breakdown, test results may help to determine the peak user load that the application is available to handle correctly using current system configurations.

№	Operation description	Time (minutes)
1	Initialize first 20 concurrent threads.	1
2	Increase load by 5 concurrent threads per 30 seconds.	20
3	Continue test execution using 250 concurrent threads.	60
4	Finish test execution, gradually stop concurrent threads.	10

10.3. Performance test scenario

Performance testing includes one test scenario that will be conducted for all test iterations. The test scenario includes the following actions:

№	Components/Applications name	Action name	Links
1	Front end	-	

Links to websites will be updated accordingly to statistical information obtained from the application. The list of websites needs to be reviewed and updated before each test iteration. If the list of websites is updated then performance scripts should be reviewed and updated accordingly before the next test iteration.

10.4. Load infrastructure

The testing team prepares load infrastructure before performance scripts execution. The infrastructure is consists of a few components:

- Load controller. The machine with installed and configured software for load testing. This station is used by the automation team to manage scripts execution, adjust the amount of virtual users during tests execution, analyze results after the testing and generate execution report.
- A set of load generators (load servers). Rented server stations for required amount of time to provide load testing. These servers are located across the world in different datacenters and used by load controller during load testing to generate virtual users (send requests to the application, process responses and collect statistics).

All servers which are the load generators are provided by Amazon EC2 cloud hosting service. The servers are located in different regions.

№	Region	Servers instance types	Amount of servers
1			
2			