Instagram

# Performance Testing Strategy

## Introduction

The purpose of this document is to specifically describe how the performance requirements for Instagram application will be tested and verified in test environment (System Under Test, SUT). 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 current document is to verify how SUT behaves under load stress. All deliverables should be used for internal (in-house) usage only and should not be presented to public.

This document describes the strategy of performance testing for the Instagram project. It consists of outlines for the following items:

* Scope of testing, test descriptions
* Non-functional requirements (NFR) related to performance
* Pass/fail criteria
* Approach
* Main scenarios
* Test data
* Requirements for test environment

## Items to be tested

Mainly, Instagram application will be tested via server side using Instagram API (based on <https://developers.facebook.com/docs/instagram> ).

|  |  |
| --- | --- |
| **#** | **Modules name** |
|  | Home Page (Feed) |
|  | Search and Recommended |
|  | Activity |
|  | Account Page |
|  | Created Posts |
|  | Created Stories |
|  | Posts and Story creation |
|  | Direct messages |

## Items not to be tested

Other features not mentioned in previous table. Client-side testing (UI) will not be performed.

## User flow

That diagram describes general flow for logged in user.

Diagram

Description automatically generated

User flow for anonymous user will not be presented, because almost all functionality is available only for logged in users.

## Application architecture

Diagram

Description automatically generated

## Approach

### Test types assumed for conducting

The following activities

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Test type** | **Demand** | **Period** |
| 1 | Smoke test | Mandatory | Regular |
| 2 | Capacity | Mandatory | After significant changes |
| 3 | Load test | Mandatory | Regular |
| 4 | Stress | Optional | Rare |
| 5 | Scalability | Optional | Rare |
| 6 | Volume | Optional | Rare |
| 7 | Durable | Optional | Rare |

1. Smoke testing

Should be performed every time when functionality of the application and the script need to be checked. Also, if needed can be used as warming up test before main testing step.

1. Capacity testing

Should be performed to find the number of virtual users which the application support in stable state. The test must be performed as one of first main tests to find optimal load model for other types of performance tests and should be performed each time after significant changes in the application or its configuration.

1. Load test

Load test is a kind of the most regular tests to check benchmark of the application and its components. Can be performed after each insignificant change in code base or configuration.

1. Stress testing

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

1. Scalability testing

Can be performed once or more in order to get multiplier(s) for different number of front-end/other servers.

1. Volume testing

Is to run with small, planned and huge amount of data with regular load to get indicators on application’s responsiveness/metrics change. Should be performed at least once or by special request.

1. Longevity testing

Supposed 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 long time running, and to check on memory leak as well.

### Level of testing

The following levels of performance testing are supposed:

|  |  |  |
| --- | --- | --- |
| **#** | **Level** | **Description** |
| 1 | Purely mocked version | Used to test only application’s tier with no engaged services: basic client-side testing, smoke testing, checking test scripts etc. |
| 2 | Partially mocked version | Used to test particular set(s) of application’s components with test data to get their benchmark with no additional load for mocked parts. |
| 3 | Full set of test data | Used for end-to-end testing of the system |

For Purely mocked version we will have mocks for all key services, such as Gateway Service, Read Service, Write Service and Synchronization Service and databases.

For Partially mocked version Write and Read Services will be used as mocks; Gateway and Synchronization Services and all databases will be used as live components.

For Full set of test data all live application services and components will be used.

For setting up of mocked services we need help from developers to create services mocks.

*Look into Application architecture section to see how services are interacting with each other.*

## Environmental needs (facilities, hardware, software, network, supplies, level of security, special tools)

### Test environment(s)

For performance testing a dedicated environment is needed. The configurations of the servers should be as much as close to production’s ones.

Expected numbers of the servers:

* Front-End servers
* Back-End servers
* Databases (PostgreSQL, Hive, Pgbouncer, Redis, Memcache)

To perform partial and full end-to-end performance testing it’s recommended to have a separated database and other services, i.e. they should not be located on the same server where related services are being tested by another team.

**Test environment specification(s):**

Close to Production environment as much as possible.

### Testing tools

| **Module** | **Software** | **Distribution** | **Where to install** | **Supported OS** | **Description** |
| --- | --- | --- | --- | --- | --- |
| **Module** | **Software** | **Distribution** | **Where to install** | **Supported OS** | **Description** |
| CI/CD | Jenkins | Manual installation | Load Generator host | Linux/ Windows | Module that starts and control execution of your performance tests |
| Monitoring agent | Telegraf | Custom installation package | All involved in testing hosts | Linux/ Windows | Module that gets all performance and health metrics from host where it installed (including Load Generator) to get more accurate picture of your performance tests impact |
| Load Generator (LG) | JMeter | Manual installation | Dedicated host | Linux/ Windows | Module that generates load on your application |
| Collector | InfluxDB | Docker image | Dedicated/Load Generator host | Linux/ Windows | Module that collects performance metrics from monitoring agents and send them to our cloud |
| Metrics visualization | Grafana | Manual installation | Dedicated host | Linux/ Windows | Module that visualizes all collected performance metrics |

## Test data

To have whole cycle of performance testing test data for Instagram should be:

1. Reusable
2. Generated in necessary amount for different stage of testing at any time:
   1. User accounts
   2. Posts with photos and videos
   3. Daily and saved stories.
   4. Some history and log records for volume testing
   5. Other
3. Cleanable (for example history, log records)

## Responsibilities

Oleksandr Maksymenko – Performance Engineer. Responsible for test script and test data creation, setting up of testing tools.

Oleksandr Maksymenko – Automation Quality Assurance Engineer. Responsible for functional validation of application.

John Doe – DevOps Engineer. Responsible for setting up of test environment, deploying of all application components to test environment.

John Doe2 – Back-End Developer. Responsible for creation of mocked services.

John Doe3 – Business Analyst (client representative). Responsible for approvals to start testing and reviews of testing results.

## Risks and contingencies

* A significant difference in configuration from the production environment
* Performance testing results can be essentially different even in case of minor difference in think times, arrival rate and test duration
* During the execution of the tests, some major performance or functional problems that may require code changes, creation of a new build may be discovered and in that case it may be necessary to repeat the load test from the beginning
* Load test should be performed against a build that is solid enough, and that has been functionally tested, after code is complete. Failure to follow this rule may result on rework to update test scripts for every new build, plus the load test may need to be repeated from the beginning. This will affect the schedule
* Performance testing tool is not capable of identically reproducing real life scenarios - so results could only be trusted as having limited reliability level
* Network/systems latency issues
* Environment’s unavailability

## Test Strategy versions

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Creator (editor)** | **Reviewer** |
| 1.0 | 22.08.2022 | Oleksandr Maksymenko | Mykhailo Kurshakov |
| 1.1 | 27.08.2022 | Oleksandr Maksymenko | Mykhailo Kurshakov |
|  |  |  |  |
|  |  |  |  |