ABN AMRO

**Functional and Load Testing Execution Report for GitLab Issues API**

horizontal line

# 

# Executive Summary

This report presents the outcomes of functional testing performed on the GitLab Issues API using Playwright and load testing conducted with k6. The primary objective was to validate the API's functionality, including issue creation, listing, updating, and deletion, and to evaluate its performance under varying loads. The tests aimed to ensure the API meets specified requirements for reliability, performance, and user experience.

## Test Environment

### Functional Testing:

Conducted on macOS ventura 13.6, using Playwright v1.12.3. Tests were automated in a Node.js environment, targeting the GitLab Issues API endpoints.

### Load Testing:

Performed using k6 v0.32.0, simulating different user loads to assess the API's performance under stress.

## Test Coverage

### Functional Tests:

* Create a new issue
* Retrieve a list of issues
* Update an issue's title and state
* Delete an issue

### Load Tests:

#### Smoke Test:

Running the following tests for 1 virtual user for a duration of 1 minute:

* Simultaneous creation of issues.
* Concurrent retrieval of issues.
* Mixed load for updating and deleting issues

#### Load Test:

Running the same set of test cases with the following configuration:

* simulate ramp-up of traffic from 1 to 30 users over 2 minutes.
* stay at 50 users for 2 minutes
* ramp-down to 0 users

#### Stress Test:

Load testing to find the maximum amount of load the system can handle with the following configuration:

* Below normal load
* Around normal load
* Ramping out to breaking point
* Beyond the breaking point
* Scale down. Recovery stage.

#### Soak Test:

load testing a system continuously and monitoring for memory leaks and behavior of the system with the following configuration:

* Ramp up to 25 users (80% of the normal capacity)
* Stay at 25 for ~4 hours
* Scale down. (optional)

## Test Execution

### Duration:

Functional testing was executed over 5 days, while load testing spanned two days.

### Test Cases Executed:

Functional: 28 test cases including positive and negative scenarios.

Load: 4 scenarios reflecting different test loads.

## 

## 

## 

## Test Outcomes

### Functional Testing Results:

Passed: 27

Failed: 1 (Issues with retrieving all the issues under all scopes)

### Load Testing Results:

Creation of issues showed response times under 2 seconds up to 100 users, with significant delays beyond 120 users.

Retrieval of issues remained stable under all tested conditions.

Updates and deletions showed increased error rates and response times at loads above 120 users.

## Defects Summary

### Functional Defects:

1 critical issue:

* The Read issue API is returning 500 internal server errors when the user tries to retrieve issues under all scopes. Ex:
* await api.listAllAvailableIssues(
* request,
* 'scope',
* 'all',
* )

### Load Test Observations:

Performance degradation and increased error rates observed beyond 120 concurrent users for creation and modification operations.

## Risk Assessment

* The functional defects identified are specific and can be mitigated with targeted fixes.
* The API shows satisfactory performance up to a moderate user load but may struggle under high traffic conditions, particularly for data creation and modification operations.

## Recommendations and Conclusion

### Immediate Actions:

Address the functional defects related to issue retrieval. Optimize API performance for handling higher loads during issue creation and updates.

### Further Testing:

Recommend conducting targeted performance optimization and re-testing. Additionally, explore implementing rate limiting or queuing mechanisms to manage high-load scenarios more effectively.

### Release Consideration:

The API is fit for release for environments expecting up to 200 concurrent users. Caution advised for high-load deployments until further optimizations and testing are completed.