DP2 2023-2024

Testing Report D04

Acme Software Factory

Software Factory

Empowering innovation through software excellence

Repository: https://github.com/rafcasceb/Acme-SF-D04

Group members:

- Flores de Francisco, Daniel danflode@alum.us.es
- Castillo Cebolla, Rafael rafcasceb@alum.us.es
- Heras Pérez, Raúl rauherper@alum.us.es
- Mellado Díaz, Luis luimeldia@alum.us.es
- Vento Conesa, Adriana adrvencon@alum.us.es

GROUP C1.049

Version 1.0

20-05-24

Content Table

Executive summary

This report will provide a comprehensive analysis of the testing procedures and results, featuring distinct sections on functional testing and performance testing. Our approach will be precise yet accessible, aiming to enhance understanding and ensure a high-quality final product.

Revision Table

Date	Version	Description of the changes	Sprint
20/05/2024	1.0	• Executive summary. • Introduction. • Functional testing.	4
24/05/2024	1.0	• Performance testing. • Conclusions.	4

Introduction

This document will provide a detailed analysis of the testing procedure and results for the following feature:

- Operations by administrators on Banners.

The content of a testing report is organized into two chapters:

- Functional testing: a listing with the test cases implemented, grouped by feature. For each test case, a succinct description plus a clear indication on how effective it was at detecting bugs are provided.

- Performance testing: it provides adequate charts, a 95%-confidence interval for the wall time taken by the application to serve the requests in the functional tests and a 95%- confidence hypothesis contrast.

Contents

Functional testing

Operations by administrator on Banners

Test case 1: list (list-all)
For this command, we just selected the button for listing several banners of the administrators many times.

For hacking, we considered accessing the URL by a wrong role. Trying to access it with a good role but a wrong user doesn't make sense (any administrator can list the banners). Finally, trying to access with an anonymous user.

It provided a coverage of 90.9 %, covering all instructions except a default assertion, which is logical. No bugs were detected.

Test case 2: show
For this command, we selected several banners using an administrator account to see their details.

For hacking, we tried accessing with a wrong role. Trying to access it with a good role but a wrong user doesn't make sense (any administrator can show the banners). Finally, we tried accessing a contract with an anonymous user.

It provided a coverage of 94.2 %, covering all instructions except a default assertion, which is logical. No bugs were detected.

Test case 3: create
For this command, we have tried to create a new banner. For each attribute we have checked the system rejects all different types of invalid data. Later, for each attribute, we have checked the system accepts all different types of valid data.

For hacking, the framework through web browser only supports to test GET hacking operations.

It provided a coverage of 93.6 %, covering all instructions except a default assertion, which is logical. No bugs were detected.

Test case 4: update
For this command, we have updated a banner of the data base. For each attribute we have checked the system rejects all different types of invalid data. Later, for each attribute, we have checked the system accepts all different types of valid data.

For hacking, the framework through web browser only supports to test GET hacking operations.

It provided a coverage of 93.6 %, covering all instructions except a default assertion, which is logical. No bugs were detected.

Test case 5: delete

For this command we tried deleting some banners using an administrator account. There are no exceptions for deleting, except for being an administrator.

For hacking, the framework through web browser only supports to test GET hacking operations.

It provided a coverage of 57.1 % since the unbind method is not executed. If we remove the unbind method it would provide an approximate coverage of 90.0% covering all instructions except a default assertion, which is logical. No bugs were detected.

Performance testing

Let us start by analyzing the performance data from the functional testing. First, we will analyze the results from the test replayer, and subsequently, we will do a hypothesis contrast.

Performance Data

The tests were performed in a laptop with an 12th Gen Intel(R) Core(TM) i7-1260P 2.10 GHz CPU and 16GB of RAM.

After performing the corresponding tests, we obtain the following figure. It represents the average response time per request path, that is, the features outlined in the previous test cases.

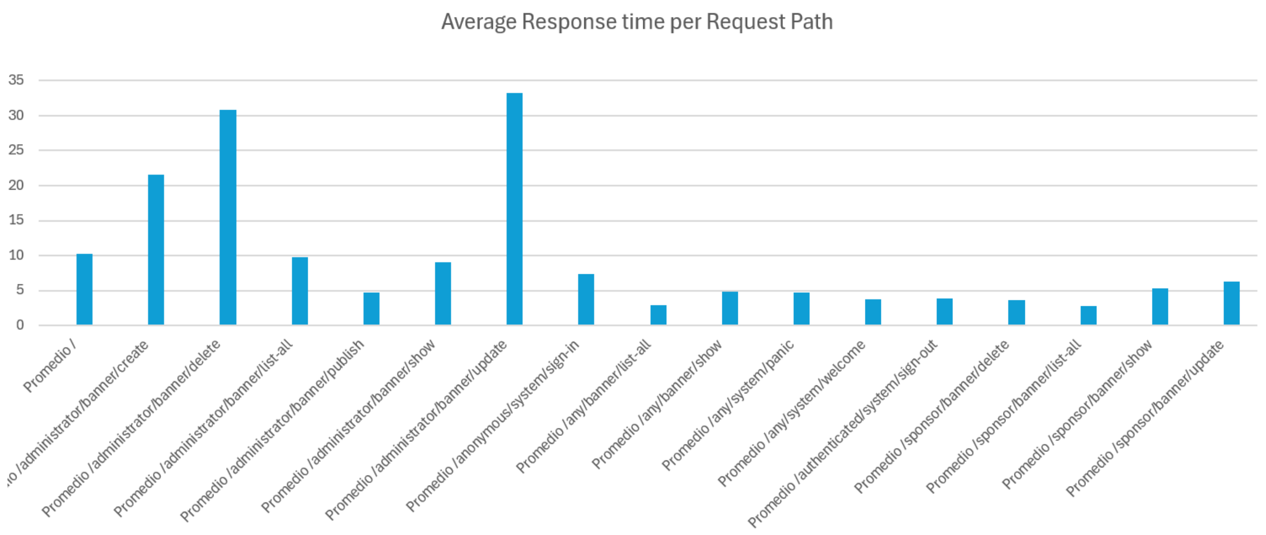

Figure 1. Average response time per request path.

As evident from our observations, the average response time across the request paths remains consistent. However, the response time increases in create, update, and delete operations. This variance is logical given the inherent simplicity of the rest of the operations, which do not complex validations and/or data binding. We consider that this variance is explicable, hence there is no need for further evaluation.

Data Analysis	
Average	15,69010673
Standard error	1,126328661
Median	8,01265
Mode	#N/D
Standard deviation	16,24414295
Sample variance	263,8721803
Kurtosis	8,226672614
Asymmetry coefficient	2,365850896

Range	107,8164
Minimum	2,4384
Maximum	110,2548
Sum	3263,5422
Count	208
Confidence level (95.0%)	2,220546132

Figure 2. Data analysis.

After computing the confidence interval, we obtain the following range: [13.46, 17.91] in milliseconds. Transformed into seconds, we obtain: [0.013, 0.017]. These ranges were obtained by computing the lower and upper limit by summing and subtracting the confidence level to the average.

We do not have any performance expectations, but the computed ranges seem acceptable, as it is below one second.

Hypothesis Contrast

We will be increasing by 10% the data obtained in Figure 2 to simulate a hypothesis contrast. Therefore, we obtain:

Data Analysis with 10% Increase	
Average	17,2591174
Standard error	1,238961527
Median	8,813915
Mode	#N/D
Standard deviation	17,86855725
Sample variance	319,2853382
Kurtosis	8,226672614
Asymmetry coefficient	2,365850896
Range	118,59804
Minimum	2,68224
Maximum	121,28028
Sum	3589,89642
Count	208
Confidence level (95.0%)	2,442600746

Figure 3. Data analysis with 10% increase in total execution time.

We obtain a new interval of: [14.81, 19.70] in milliseconds, and [0.014, 0.019] seconds.

We will now use a Z-Test to compare the results.

	time	*increased*
Average	15,69010673	17,2591174
Variance (known)	263,8721803	319,2853382
Data points	208	208

Hypothesized difference of means	0
z	-0,937054015
P(Z<=z) one-tail	0,174365387
Critical value of z (one-tail)	1,644853627
Critical value of z (two-tail)	0,348730774
Critical value of z (two-tail)	1,959963985

Figure 4. Z-test for two sample means.

To interpret the Z-Test outcome, we examine the p-value. This value is computed assuming a two-tailed test. Alpha will be 0.05, as we indicated it when performing the analysis.

Next, we compare this two-tailed p-value to alpha. If the p-value exceeds alpha, it suggests that the observed changes did not yield significant improvements. In our case, the critical value of z (two-tail) is approximately 0.34, notably higher than our selected alpha level. This suggests that the observed difference in averages lacks statistical significance, indicating that our performance did not improve. This outcome aligns with expectations since we introduced a 10% increase in time from the initial measurement.

Conclusions

In short, the performance data gathered from the functional testing indicates that the system operates well within expected parameters. While we did observe minor increases in response times, particularly in simpler tasks, these were anticipated and do not raise any significant concerns.

Bibliography

Intentionally blank.