20 DE MAYO DE 2024

TESTING REPORT STUDENT #1

GROUP C-1.047

Acme SF, Inc.

Repository link: https://github.com/Cargarmar18/Acme-SF

Carlos García Martínez
Email: cargarmar18@alum.us.es

Table of contents

1. Executive summary

The functional testing chapter provides a detailed listing of tests implemented by feature, each with a complete description of the function. Overall, the functional tests were highly effective, revealing mainly minor bugs that significantly enhanced the application.

The performance testing chapter presents an analysis of the time taken by the project to serve requests (being one 10% more than the other) on a computer, including detailed charts and 95%-confidence intervals. These results are crucial for optimizing and analyzing the performance of the system.

2. Revision table

Revision number	Date	Description
1.0.0	20-05-2024	Creation and development of the document.
1.0.1	25-05-2024	Update and correction of data

3. Introduction

This document provides a comprehensive analysis of the testing and performance evaluation of the mandatory features developed by Student #1. It aims to ensure that the implemented functionalities meet the required standards of correctness, security, and efficiency. By systematically examining each test case and performance metric.

The analysis is divided into two main sections: Functional Testing and Performance Testing. The Functional Testing section is further broken down into two subsections. The first subsection, Test Case Analysis, delves into the specifics of individual test cases for various features such as project and user story management. Each test case is described in detail, outlining the scenarios tested, the methodology used, and the effectiveness in identifying bugs. This includes testing both normal and the limit cases, as well as security measures to ensure that only authorized roles can perform certain actions.

The second subsection, Coverage Analysis, provides a quantitative assessment of the test coverage across different classes. It includes a detailed table showing the coverage percentages for various services related to project and user story management.

The Performance Testing section focuses on evaluating the efficiency of the system by analyzing the time it takes to perform various manager-related actions. The analysis identifies the most time-consuming actions and provides insights into why certain operations are less efficient.

Further, the Confidence Interval subsection presents the computations for the 95%-confidence interval for the time taken by the system to serve requests. The document also includes a Hypothesis Contrast subsection that simulates a performance improvement scenario by assuming a 10% increase in efficiency. Through Z-testing analysis, the document evaluates whether the observed performance changes are statistically significant.

4. Contents

4.1Testing analysis:

In this section, I will analyze the test suite generated for the mandatory features of Student #1. This analysis is divided into two parts: the first part focuses on examining each individual test case, while the second part addresses the coverage analysis.

4.1.1 Test case analysis:

- List-mine project:

This test case involves accessing the project listings of a manager to verify if the feature is correctly implemented. This was done by interacting with the listing of the projects of that manager.

Additionally, to test the security of this feature, I attempted to access the listings without a manager role. As expected, I received an error indicating that other roles are not authorized.

- Show project:

This test case involves accessing the specific data of a manager's project to verify if the feature is correctly implemented. This was done accessing multiple data of multiple projects from that manager.

Additionally, to test the security of this feature, I attempted to access the show without a manager role and accessing the data of that project with another manager (right role, wrong user) and testing for an id that doesn't correspond to a project. As expected, I received an error indicating that other roles are not authorized and that user neither.

- Create project:

This test case involves creating a project as a manager to verify if the feature is correctly implemented. This was done by trying many combinations in the form to check if it doesn't panic when sending an empty form or wrong values (including the custom constraints) and

checking how the feature handles conflictive values such as the first and last possible value, SQL injection…

Additional testing for hacking cannot be done due to the framework not supporting POST hacking testing.

- Delete project:

This test case involves deleting a specific project of a manager to verify if the feature is correctly implemented. This was done by interacting with the specific project and deleting it if it is in draft mode.

Additional testing for hacking cannot be done due to the framework not supporting POST hacking testing.

- Update project:

This test case involves updating a project as a manager to verify if the feature is correctly implemented. This was done by trying many combinations in the form to check if it doesn't panic when sending an empty form or wrong values (including the custom constraints, highlighting that the code during the update has an specific custom constraint since it can retain the pre-update code which is "taken" in the database) and checking how the feature handles conflictive values such as the first and last possible value, SQL injection…

Additional testing for hacking cannot be done due to the framework not supporting POST hacking testing.

- Publish project:

This test case involves publishing a project as a manager to verify if the feature is correctly implemented. This was done by trying many combinations with the constraints implemented so it can be checked: checking that I am not allowed to publish a project when it has no user story, checking that I cannot publish a project with at least one of its user stories unpublished and checking that I cannot publish a project with fatal errors.

Additional testing for hacking cannot be done due to the framework not supporting POST hacking testing.

- List-mine user story:

This test case involves accessing the user story listings of a manager to verify if the feature is correctly implemented. This was done by interacting with the listing of the user stories of that manager.

Additionally, to test the security of this feature, I attempted to access the listings without a manager role. As expected, I received an error indicating that other roles are not authorized.

- List user story:

This test case involves accessing the user story listings of a project from a manager to verify if the feature is correctly implemented. This was done by interacting with that specific listing of the user stories.

Additionally, to test the security of this feature, I attempted to access the show without a manager role and accessing the data of that listing with another manager (right role, wrong user) and testing for an id that doesn't correspond to an user story. As expected, I received an error indicating that other roles are not authorized and that user neither.

- Show user story:

This test case involves accessing the specific data of a manager's user story to verify if the feature is correctly implemented. This was done accessing multiple data of multiple user stories from that manager.

Additionally, to test the security of this feature, I attempted to access the show without a manager role and accessing the data of that project with another manager (right role, wrong user) and testing for an id that doesn't correspond to a user story. As expected, I received an error indicating that other roles are not authorized and that user neither.

- Create user story:

This test case involves creating a user story as a manager to verify if the feature is correctly implemented. This was done by trying many combinations in the form to check if it doesn't panic when sending an empty form or wrong values (including the custom constraints) and checking how the feature handles conflictive values such as the first and last possible value, SQL injection…

Additional testing for hacking cannot be done due to the framework not supporting POST hacking testing.

- Delete user story:

This test case involves deleting a specific user story of a manager to verify if the feature is correctly implemented. This was done by interacting with the specific project and deleting it if it is in draft mode.

Additional testing for hacking cannot be done due to the framework not supporting POST hacking testing.

- Update user story:

This test case involves creating a user story as a manager to verify if the feature is correctly implemented. This was done by trying many combinations in the form to check if it doesn't panic when sending an empty form or wrong values (including the custom constraints) and checking how the feature handles conflictive values such as the first and last possible value, SQL injection…

Additional testing for hacking cannot be done due to the framework not supporting POST hacking testing.

- Publish user story:

This test case involves publishing a user story as a manager to verify if the feature is correctly implemented. This was done by publishing user stories and check that the feature works.

Additional testing for hacking cannot be done due to the framework not supporting POST hacking testing.

- Create user story project:

This test case involves creating a user story project as a manager, that's it, linking a user story to a given project to verify if the feature is correctly implemented. This was done by checking when sending an empty formulary and by adding user stories to a specific project inside the "user stories" tab of the project.

Additionally, to test the security of this feature, I attempted to access the listings feature for the user stories of a project without a manager role and with another manager (right role,

wrong user). As expected, I received an error indicating that other roles are not authorized and that user neither.

- Delete user story project:

This test case involves creating a user story project as a manager, that's it, linking a user story to a given project to verify if the feature is correctly implemented. This was done by checking when sending an empty formulary and by adding user stories to a specific project inside the "user stories" tab of the project.

Additionally, to test the security of this feature, I attempted to access the listings feature for the user stories of a project without a manager role and with another manager (right role, wrong user). As expected, I received an error indicating that other roles are not authorized and that user neither.

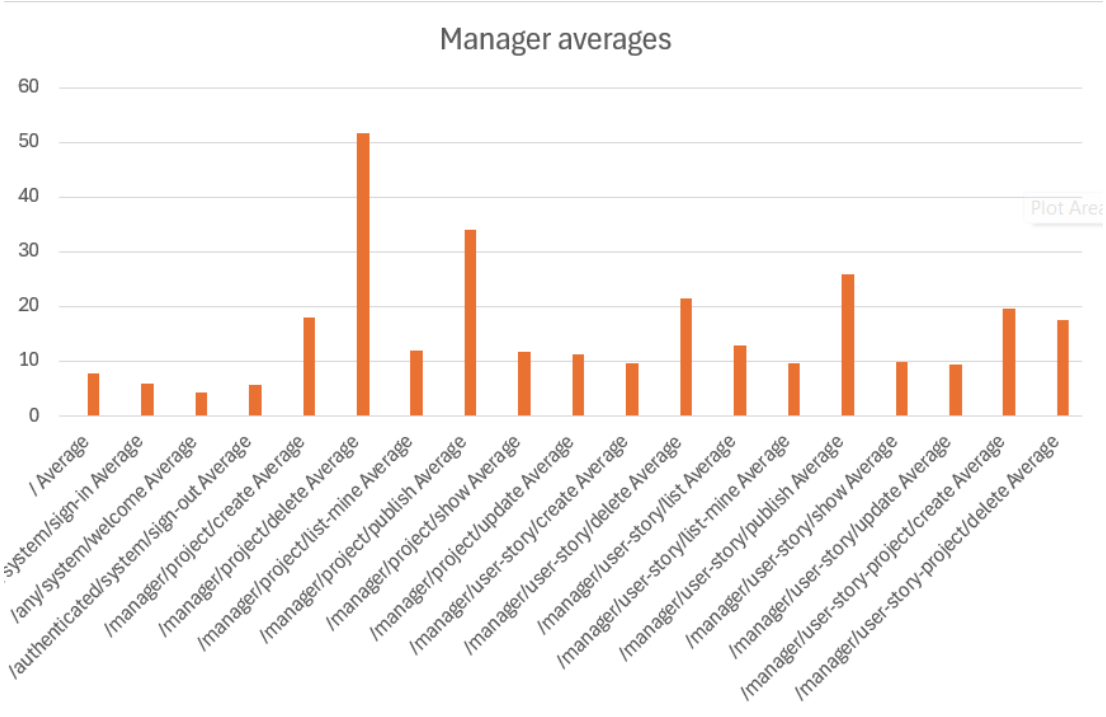
4.1.2 Coverage analysis:

Class Name	Coverage
ManagerUserStoryDeleteService	90,0%
ManagerUserStoryPublishService	89,8%
ManagerUserStoryCreateService	89,2%
ManagerUserStoryUpdateService	90,4%
ManagerUserStoryListMineService	93,1%
ManagerUserStoryShowService	96,7%
ManagerUserStoryListService	94,3%
ManagerProjectDeleteService	90,1%
ManagerProjectPublishService	92,8%
ManagerProjectCreateService	91,4%
ManagerProjectUpdateService	89,9%
ManagerProjectListMineService	93,5%
ManagerProjectShowService	96,5%
ManagerUserStoryProjectCreateService	91,9%
ManagerUserStoryProjectDeleteService	92,2%

Providing the table with the data, we can conclude that the coverage is overall complete knowing that the non-covered code, is mainly code provided by the ACME framework or unreachable.

4.2Performance analysis:

After executing the tests, the following chart shows the results of the average performance of each manager related action.



Looking into this graph we can conclude that the most time-consuming action is the deletion of a project. This makes sense when analyzing that the feature also must perform the elimination of each user story connection to that project. Overall, the project that implements user story project actions are the most inefficient ones while the listing, updating and creation of user story or project are the most efficient.

- Confidence interval:

Let us now see the computations for the confidence interval:

Column1				
		Interval (ms)	10,17954252	11,72997192
Mean	10,95475722	Interval (s)	0,010179543	0,011729972
Standard Error	0,394884121			
Median	9,25045			
Mode	10,1707			
Standard Deviation	10,79991827			
Sample Variance	116,6382346			
Kurtosis	186,3176682			
Skewness	11,33212832			
Range	210,1749			
Minimum	2,3055			
Maximum	212,4804			
Sum	8194,1584			
Count	748			
Confidence Level(95,0%)	0,7752147			

Even if we don't have any specific requirement for the confidence interval in our project, a confidence interval of [0,010 , 0,011] assuming the 95% confidence level is an overall great interval time.

- Hypothesis contrast:

We don't have any requirement regarding the performance of the system and indexes are already implemented so to simulate a hypothesis contrast let us assume an increase of 10% over the time.

Keeping this in mind, this is the confidence interval of the generated data in contrast.

Before			*After*		
Mean	10,95475722		Mean	12,05023294	
Standard Error	0,394884121		Standard Error	0,434372533	
Median	9,25045		Median	10,175495	
Mode	10,1707		Mode	11,18777	
Standard Deviation	10,79991827		Standard Deviation	11,87991009	
Sample Variance	116,6382346		Sample Variance	141,1322638	
Kurtosis	186,3176682		Kurtosis	186,3176682	
Skewness	11,33212832		Skewness	11,33212832	
Range	210,1749		Range	231,19239	
Minimum	2,3055		Minimum	2,53605	
Maximum	212,4804		Maximum	233,72844	
Sum	8194,1584		Sum	9013,57424	
Count	748		Count	748	
Confidence Level(95,0%)	0,7752147		Confidence Level(95,0%)	0,85273617	
Interval(ms)	10,17954252	11,72997192	Interval(ms)	11,19749677	12,90296911
Interval(s)	0,010179543	0,011729972	Interval(s)	0,011197497	0,012902969

Now, the Z-testing analysis for both cases is as follows:

z-Test: Two Sample for Means		
	Before	*After*
Mean	10,95475722	12,05023294
Known Variance	116,6382346	141,1322638
Observations	748	748
Hypothesized Mean Difference	0	
z	-1,866108755	
P(Z<=z) one-tail	0,031013076	
z Critical one-tail	1,644853627	
P(Z<=z) two-tail	0,062026151	
z Critical two-tail	1,959963985	

To analyze the impact of the change it's needed to pay attention to the "P(Z<=z) two-tail" row. It shows a value of 0,06, with this in mind and knowing that α = 1 - confidence level = 0,05, Since P(z<=z) is above 0,05 and is not a value really close to 0,05 (as for example 0,0500001 would be) we can ensure that the changes didn't affect to our performance, the sample times are different but globally they are the same.

5. Conclusions

In conclusion, the thorough testing and performance analysis conducted on the mandatory features developed by Student #1 provide valuable insights into the efficiency of the system. Through examination of test cases, we have ensured that critical functionalities such as project and user story management meet the required standards of correctness and security.

As for the performance identified key areas of optimization. The observed variations in performance metrics, with the computation of confidence intervals and hypothesis contrasts, offer a huge understanding of the system's efficiency under different scenarios.

6. Bibliography

Intentionally blank