Insertando imagen...

ACME-SF

G1.007

**Testing report S05-D04**

26/05/2024



# Cover

|  |  |
| --- | --- |
| **Repository:** <https://github.com/Pablo-Caballero-Maria/Acme-One-24.1.0-C1.07> | |
| **Student #1**  **ID:** 31878881F  **UVUS:**  pabcabmar3  **Name:** Caballero María, Pablo **Roles:** manager, developer | **Student #2**  **ID Number:**49034820Q  **UVUS:** mararnmon  **Name:** Arnáiz Montero, Marco Antonio  **Roles:** developer, operator |
| **Student #3**  **ID Number:** 77865211E  **UVUS:**  alfalolan  **Name:** Alonso Lanzarán, Alfonso Luis  **Roles:** developer, tester | **Student #4**  **ID Number:** 53932912M  **UVUS:** albsanmim  **Name:** Sánchez Mimbrero, Alberto  **Roles:** developer |
| **Student #5**  **ID Number:** 48123111G  **UVUS:** juagarcar4  **Name:** Garcia Carballo, Juan  **Roles:** developer, analyst, tester |  |

Table of contents

[Cover 2](#_Toc167711670)

[Executive summary 4](#_Toc167711671)

[Revision table 4](#_Toc167711672)

[Introduction 4](#_Toc167711673)

[Contents 4](#_Toc167711674)

[Functional Testing 5](#_Toc167711675)

[Code Audit 5](#_Toc167711676)

[Audit Record 6](#_Toc167711677)

[Sample data used for create, update and publish tests. 7](#_Toc167711678)

[Table with all traces make for both funtionality: 9](#_Toc167711679)

[Performance Testing 9](#_Toc167711680)

[ Statistical analysis 9](#_Toc167711681)

[ Statistical metrics and intervals: 11](#_Toc167711682)

[ Profiling 13](#_Toc167711683)

[Conclusions 13](#_Toc167711684)

[Bibliography 14](#_Toc167711685)

# Executive summary

This document serves as a comprehensive testing report for project S05-D04. The report details the testing processes, methodologies, and results for verifying the requirements and functionality of the entire project. The primary focus of this report is on the testing tasks executed by the student team, specifically those carried out by student #5. The testing activities included functional testing, code audits, performance testing, and a thorough statistical analysis to ensure the application meets the expected standards and performs reliably under various conditions.

# Revision table

|  |  |  |
| --- | --- | --- |
| Number | Date(dd/mm/yyyy) | Description |
| 1.0 | 26/05/2024 | Document done in its entirety, reviewed by peers. No major errors were found. |

# Introduction

The objective of this testing report is to provide a detailed account of the testing activities conducted for project S05-D04. The testing process encompassed various aspects, including functional testing, code audits, and performance testing.

Each functionality was rigorously tested to identify and rectify potential bugs, ensuring the application meets its requirements. The results from these tests showed that the application functions correctly and securely, with all identified issues being resolved. Performance testing revealed that the implementation of indexes had minimal impact on the average response times, confirming the application’s efficiency and reliability.

# Contents

## Functional Testing

### Code Audit

• List-mine

To fully test the functionality of list-mine, we have tried to obtain a list with multiple code-audits and an empty list without any code-audit. To test if it behaved correctly, we performed some hacking tests. In this case, a wrong role test was conducted, where it checks that the user has the auditor role to list the code audits, and if incorrect, the application throws a 500 error (panic error).

We don’t find any bug.

• Show

To test the show functionality, a ”.safe” type test was carried out, where different code-audits were shown to verify that their data is displayed correctly. To test if it behaves correctly, hacking tests were conducted. In this case, a test was done to check if the user has the auditor role and another test to check if the code-audit to be displayed belongs to the user who made the request. If any of these checks fail, the application should throw a ‘panic error’ (500 error).

We don’t find any bug.

• Create

To fully test the functionality of creating a code-audit, two .safe type tests were created. The first one checks a large amount of test data in the form to ensure that invalid data is not allowed and to verify that the application shows the correct errors. The second test was used to add more data to the test, as the first test did not correctly test the “optionalLink” attribute. Additionally, .hack type tests were conducted to verify that the application throws a “panic error” when trying to create a code-audit with a user who does not have the auditor role.

We don’t find any bug.

• Update

To fully test the functionality of updating a code-audit, a .safe type test was created. As with the “create” functionality, it was checked that the application does not accept incorrect data and throws the appropriate errors by testing a large amount of test data. Additionally, three .hack type tests were conducted to verify that the application throws a “panic error” when attempting to modify a code-audit without having the auditor role, without owning the code-audit to be modified, or trying to modify an already published code-audit. In all these cases, the application throws a 500 error.

We don’t find any bug.

• Delete

To fully test the functionality of deleting a code-audit, a .safe type test was created where it was verified that different code-audits can be deleted regardless of whether they have associated audit-records or not. Additionally, three .hack type tests were conducted to verify that the application throws a “panic error” when attempting to delete a code-audit without having the auditor role, without owning the code-audit to be deleted, or trying to delete an already published code-audit. In all these cases, the application throws a 500 error.

We don’t find any bug.

• Publish

To fully test the functionality of publishing a code-audit, a .safe type test similar to “update” or “create” was created, where through a large amount of data it was verified that the application throws the expected errors and only allows data that meets the requirements. Additionally, three .hack type tests were conducted to verify that the application throws a “panic error” when attempting to publish a code-audit without having the auditor role, without owning the code-audit to be published, or trying to publish an already published code-audit. In all these cases, the application throws a 500 error.

We don’t find any bugs.

### Audit Record

* List-Mine

To fully test the functionality of list-mine, we have tried to obtain a list with multiple audit-records for various different code-audits as well as an empty list with no elements. To test if it behaved correctly, we performed some hacking tests. In this case, a wrong role test was conducted to check if the user has the auditor role to list the audit-records, and another test to verify that the code-audit of the audit-record belongs to the user. If any of these tests fail, the application throws a 500 error (panic error).

Thanks to testing the hacking cases, it was discovered that when a user who was not the creator of the code-audit whose audit-records were listed tried to list them, they were displayed. This was a breach in the security of the application and we were able to find and resolve it thanks to the tests.

* Show

To test the show functionality, a .safe type test was conducted where different audit-records were displayed to verify that their data is shown correctly. To test if it is not susceptible to hacking, hacking tests were performed. In this case, one test checks if the user has the auditor role, and another test checks if the code-audit to be shown belongs to the user who made the request. If any of these checks fail, the application should throw a ‘panic error’ (500 error).

We don’t find any bugs.

* Create

To fully test the functionality of creating an audit-record, one .safe type test was created. This test checks through a large amount of test data in the form to see if the application accepts the data that meets the requirements and throws the corresponding error when the data is invalid. Additionally, .hack type tests were conducted to verify that the application throws a “panic error” when attempting to create an audit-record with a user who does not have the auditor role or when the user does not own the code-audit to which the audit-record is being added.

Thanks to generating the tests, we discovered that despite the code-audit not being owned by the user, a user could still create audit-records for it. This resulted in a security breach in the application. Additionally, no error was shown when entering a period where the end of the period occurred before the start of the period. Thanks to generating the tests, we were able to locate and fix these bugs in the application's functionality.

* Update

To fully test the functionality of updating an audit-record, a .safe type test was created. In this test, as in the “create” functionality, it was checked that the application does not accept incorrect data and throws the appropriate errors by testing a large amount of test data. Additionally, three .hack type tests were conducted to verify that the application throws a “panic error” when attempting to modify an audit-record without having the auditor role, without owning the audit-record to be modified, or trying to modify an already published audit-record. In all these cases, the application throws a 500 error.

Thanks to generating the test, we discovered that no error was shown when entering a period where the end of the period occurred before the start of the period. Thanks to generating the tests, we were able to locate and fix these bugs in the application's functionality.

* Delete

To fully test the functionality of deleting an audit-record, a .safe type test was created where it was verified that different audit-records can be deleted regardless of whether they have associated audit-records or not. Additionally, three .hack type tests were conducted to verify that the application throws a “panic error” when attempting to delete an audit-record without having the auditor role, without owning the audit-record to be deleted, or trying to delete an already published audit-record. In all these cases, the application throws a 500 error.

We don’t find any bugs.

* Publish

To fully test the functionality of publishing an audit-record, a .safe type test similar to “update” or “create” was created, where through a large amount of data it was verified that the application throws the expected errors and only allows data that meets the requirements. Additionally, three .hack type tests were conducted to verify that the application throws a “panic error” when attempting to publish an audit-record without having the auditor role, without owning the audit-record to be published, or trying to publish an already published audit-record. In all these cases, the application throws a 500 error.

Thanks to generating the test, we discovered that no error was shown when entering a period where the end of the period occurred before the start of the period. Thanks to generating the tests, we were able to locate and fix these bugs in the application's functionality.

### Sample data used for create, update and publish tests.

This are the data used for the positives and negatives test cases in the create, update and publish test.

In these cases, boundary cases are tested, i.e, the lower/upper limit minus one, the lower/upper limit, and the lower/upper limit plus one. Additionally, an intermediate case and a case where the input field is empty. For dates, we also test a leap year case, for links, we test all their variants and for strings we try sql and script injection in top of non-latin characters.

Here are screenshots of the excel sheets that we used to make the application tests as complete as possible:

* Code-audit:

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

* Audit Record:

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

### Table with all traces make for both funtionality:

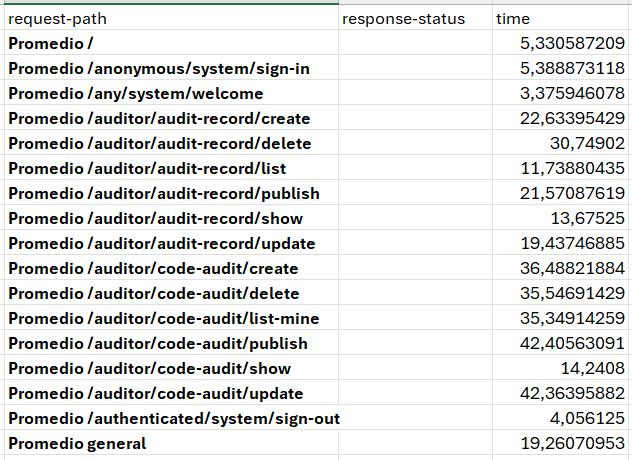
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **CODE AUDITS** | | | | | | **AUDIT RECORD** | | | | | |
|  | **LIST** | **SHOW** | **CREATE** | **UPDATE** | **DELETE** | **PUBLISH** | **LIST** | **SHOW** | **CREATE** | **UPDATE** | **DELETE** | **PUBLISH** |
| SAFE | X | X | X | X | X | X | X | X | X | X | X | X |
| WRONG ROLE | X | X | X | X | X | X | X | X | X | X | X | X |
| WRONG USER |  | X |  | X | X | X | X | X | X | X | X | X |
| ITS PUBLISHG |  |  |  | X | X | X |  |  |  | X | X | X |

## Performance Testing

### Statistical analysis

After obtaining a large number of test traces through functional testing. I analyze the performance of the application. First I launched the “tester#player” and using the Excel tools I was able to analyze the average time of the functionalities.

* Performance Charts:
  + Before implementing indexes



Gráfico, Gráfico de barras

Descripción generada automáticamente

* After implementing indexes

Tabla

Descripción generada automáticamente

Gráfico, Gráfico de barras

Descripción generada automáticamente

We can observe that after adding indexes, there have not been significant changes. In fact, we can observe that the average time has slightly increased in some queries (even in the base project calls such as */anonymous/system/sign-in*). This may be caused because at the time of executing the tests they were done on different days, and during the execution of the index tests there were other applications in the background that worsened the performance. Even so, the change is so minuscule that it is irrelevant, since in the general average it increases only 1 millisecond compared to the test execution without index.

### Statistical metrics and intervals:

* + Before implementing indexes

Tabla

Descripción generada automáticamente

* + After implementing indexes

Tabla

Descripción generada automáticamente

* + Analysis:

If we compare both races, we can see that the second race is a little worse than the first, but seeing that on average it is 1 millisecond slower we can take it as irrelevant. This difference in times may be caused because at the time of executing the tests they were done on different days, and during the execution of the index tests there were other applications in the background that worsened the performance. Even so, the change is so minuscule that it is irrelevant, since in the general average it increases only 1 millisecond compared to the test execution without index.

Computing both tails z values based on our test data, we obtain:

Texto, Tabla

Descripción generada automáticamente

Here, we can see how the z-value is above the ideal range, with the maximum being the alpha value (0.05). This means that our change adding the indexes was irrelevant and no notable changes were made in the performance.

### Profiling

* + Software profiling

Monitoring the CPU time consumed during the replaying of the test using VisualVM. We can find the hotspots on the code, we sorted it by CPU time:

Tabla

Descripción generada automáticamente

The 2 biggest time consuming hotspots are not implemented by me.

However the unbind method for list-mine consume 10 miliseconds, that means that it’s the unbind who is consuming time, this may have been caused by the fact that in the list the code-audit notes must be calculated for each one, as specified in the forum (explain in the analysis report of the third deliverable), this could cause an increase in the time it takes to execute.

The next 10 methods don’t have self time, instead they have total time (between 0.1 and 0.6 seconds), this is caused by calls to their ultimate methods that they invoke from the framework.

# Conclusions

In conclusion, the testing activities have successfully validated the application's functionality, security, and performance. The comprehensive testing process, including both functional and performance testing, ensured that all critical aspects of the application were thoroughly examined. Any identified bugs or security breaches were promptly addressed, resulting in a robust and reliable application.

The performance analysis demonstrated that the application maintains its efficiency even after implementing indexes. This report confirms that the project meets its intended requirements and is ready for deployment, ensuring a secure and efficient user experience.

# Bibliography

Intentionally Blanck