Un dibujo con letras

El contenido generado por IA puede ser incorrecto.

TESTING report

Repository: https://github.com/C1-036/Acme-ANS-C2

Group: C2.036

Sevillano Barea, Alejandro - [alesevbar@alum.us.es](mailto:alesevbar@alum.us.es) Naredo Bernardos, Ignacio- [ignnarber@alum.us.es](mailto:ignnarber@alum.us.es)

Escudero Aldana, David - [davescald@alum.us.es](mailto:davescald@alum.us.es)

July 04, 2025

Content Table

[1. Executive Summary 2](#_Toc202485528)

[2. Revision Table 2](#_Toc202485529)

[3. Introduction 2](#_Toc202485530)

[4. Contents 3](#_Toc202485531)

[4.1 Funcional Testing 3](#_Toc202485532)

[4.2 Performance Testing 10](#_Toc202485533)

[5. Conclusions 16](#_Toc202485534)

[6. Bibliography 17](#_Toc202485535)

# 1. Executive Summary

This report presents the functional and performance evaluation of our project. It is structured into two main sections: Functional Testing and Performance Testing.

The Functional Testing section details a complete set of test cases, categorized by functionality. Each case includes a brief description and an analysis of its effectiveness in detecting errors. This methodical approach allows for comprehensive coverage of all system functions, using Eclipse and Excel to facilitate execution.

The Performance Testing section offers a comprehensive analysis based on graphs and statistical data. Response times for processing requests during functional testing were evaluated on two different machines, calculating 95% confidence intervals. A hypothesis test was also performed with the same confidence level to determine which machine offers better performance. Furthermore, the system's performance was compared before and after adding indexes to the entities, again using a hypothesis test with a 95% confidence level.

The objective of this report is to provide a clear and structured view of the project's robustness and efficiency, ensuring that all essential aspects are rigorously tested and analyzed.

# 2. Revision Table

|  |  |  |
| --- | --- | --- |
| Revision number | Date | Description |
| 1 | 01/07/2025 | Creation of the report |
| 2 | 04/07/2025 | Completed report |

# 3. Introduction

This report presents the results of a comprehensive project testing process, encompassing both functional and performance testing. The objective of these tests is to ensure the robustness, security, and efficiency of the system. Functional testing focuses on verifying the correctness of functionalities, while performance testing evaluates the system's ability to efficiently handle requests under various conditions.

**Functional Testing Summary**

The functional testing chapter describes the execution of various test cases designed to validate the behavior of the system's functionalities. These cases are categorized into positive, negative, and simulated attack scenarios for a comprehensive assessment. The main objective is to detect vulnerabilities and confirm the system's resilience to both expected and unexpected conditions. Each test case includes its purpose, the manner in which it was executed, and its effectiveness in identifying errors.

**Performance Testing Summary**

The performance testing chapter analyzes system response times on two different computers using a suite of functional tests. This analysis is supported by detailed graphs and includes a 95% confidence interval for processing times. In addition, a hypothesis test is performed to determine which hardware offers the best performance and to evaluate the impact of adding indexes to the database tables on overall system performance.

The document is structured to provide a clear and organized review of the testing process and its results. It begins with an executive summary highlighting the most relevant findings and recommendations. This is followed by the main sections: functional testing and performance testing, each with specific sections on the scope, methodology, results, and key findings. Finally, the conclusions are presented, summarizing the most important points, followed by an empty bibliography as no relevant external sources were used.

# 4. Contents

## Funcional Testing

**Introduction to Functional Testing**

The functional tests performed in this project were carefully executed to ensure both the robustness and security of the developed functions. The process included a variety of test cases, covering both expected and unexpected situations, as well as attack simulations, with the purpose of detecting potential vulnerabilities in the system.

The test cases were organized into the following categories:

* Positive and negative test cases: Stored in files with a. safe extension in the src/test/resources directory, these cases were designed to test how the system responds to normal conditions and unexpected behaviors.
* Hacking test cases: Saved in .hack files, they were created specifically to try to push the system to its limits and reveal potential weaknesses that could be exploited by malicious users.

All implemented functionalities were evaluated through these tests, rigorously following the guidelines established in the "S01 - Formal Testing" document.

**Test Cases Implemented**

Flight Test Cases

Test cases 1 – Create:

**Tests.safe**

* Create a Flight with empty data.
* Create a Flight with various erroneous values in its attributes.
* Create a Flight with valid and invalid boundary values for its attributes.
* Create a Flight with a variety of valid values in its attributes.

**Tests.hack**

* Attempt to create a Flight while logged in but with a role other than manager.

Test cases 2 – List:

**Tests.safe**

* List all Flights that belong to a manager.

**Tests.hack**

* Attempt to access a list of Flights while logged in with a role other than manager.

Test cases 3 – Show:

**Tests.safe**

* Display the data of a Flight that belongs to a manager.

**Tests.hack**

* Attempt to access a Flight while logged in with a role other than manager.
* Attempt to access a Flight while logged in as a manager, but accessing one that does not belong to them.
* Attempt to retrieve the data of a non-existent Flight.

Test cases 4 – Update:

**Tests.safe**

* Update a Flight with empty data.
* Update a Flight with various erroneous values in its attributes.
* Update a Flight with valid and invalid boundary values for its attributes.
* Update a Flight with a variety of valid values in its attributes.

**Tests.hack**

* Attempt to update a Flight while logged in but with a role other than manager.
* Attempt to update a Flight while logged in as a manager, but for a Flight that does not belong to them.
* Attempt to update a Flight while logged in as a manager, but for a published Flight.
* Attempt to update a Flight while logged in as a manager, but trying to edit read-only attributes that should be ignored by the framework.

Test cases 5 – Delete:

**Tests.safe**

* Delete a Flight that belongs to the manager.
* Attempt to delete a Flight when it is assigned(“Flightassignment”).

**Tests.hack**

* Attempt to delete a Flight while logged in with a role other than manager.
* Attempt to delete a Flight while logged in as a manager, but for a Flight that does not belong to them.
* Attempt to delete a Flight while logged in as a manager, but for a published Flight.
* Attempt to delete a Flight while logged in as a manager, but for a Flight that does not exist.

Test cases 6 – Publish:

**Tests.safe**

* Publish a Flight that belongs to the manager and whose segments are already published.
* Attempt to publish a Flight that has no segments.
* Attempt to publish a Flight without all its segments being published.
* Attempt to publish a Flight for which no auto-transfer has been configured and whose segments use different aircraft.

**Tests.hack**

* Attempt to publish a Flight while logged in with a role other than manager.
* Attempt to publish a Flight while logged in as a manager, but for a Flight that does not belong to them.
* Attempt to publish a Flight while logged in as a manager, but for a Flight that does not exist.
* Attempt to publish a Flight while logged in as a manager, but for a Flight that is already published.

**Cobertura en base a los servicios de Flight**

A screenshot of a computer

AI-generated content may be incorrect.

In this section dedicated to the test cases for flights, we have achieved extensive coverage with the previously mentioned test cases, reaching an overall final coverage of 99.9% in the flight features. Notably, a code refactoring was undertaken to improve the coverage. This enhancement demonstrates a commitment to code quality and thorough testing, ensuring that most of the code is adequately evaluated and validated.

Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

The createService doesn’t reach 100% coverage because of that condition, which doesn’t really make sense—authorise() only checks that the active principal isn’t null, and in practice, on every valid execution path getRequest().getPrincipal().getActiveRealm() never returns null. You shouldn’t use that authorise() at all, but this one instead:

public void authorise() {

super.getResponse().setAuthorised(true);

}

I haven’t added it because I’d have to re-run the create tests, but that would be the solution.

A close-up of text

AI-generated content may be incorrect.

Since there are no tests simulating those three cases (null flight, different manager, or draft mode disabled), the tool highlights the method in yellow to indicate that some code paths are not covered.

Leg Test Cases

Test cases 1 – Create:

**Tests.safe**

* Create a Leg with empty data.
* Create a Leg with a variety of erroneous values in its attributes.
* Create a Leg with valid and invalid boundary values for its attributes.
* Create a Leg with a variety of valid values in its attributes.
* Create a Leg to demonstrate the validation that the scheduled departure cannot be in the past.
* Create a Leg to demonstrate the validation that the arrival must be set in the future.
* Create a Leg to demonstrate the validation that the departure and arrival airports cannot be the same.
* Create a Leg to demonstrate the validation that the departure airport does not match the arrival airport of the previous segment.

**Tests.hack**

* Attempt to create a Leg while logged in with a role other than manager.
* Attempt to create a Leg while logged in as a manager, but for a Flight that does not belong to them.
* Attempt to create a Leg while logged in as a manager, but when the Flight is already published.
* Attempt to create a Leg while logged in as a manager, but trying to create it with read-only attributes that should be ignored by the framework.
* Attempt to create a Leg while logged in as a manager, but trying to create it with navigation attributes by providing illegal values.

Test cases 2 – List:

**Tests.safe**

* List all Legs that belong to a manager.

**Tests.hack**

* Attempt to access a list of Legs while logged in with a role other than manager.
* Attempt to access a list of Legs while logged in as a manager, but for a Flight that does not belong to them.

Test cases 3 – Show:

**Tests.safe**

* Display the data of a Leg that belongs to a manager.

**Tests.hack**

* Attempt to access a Leg while logged in with a role other than manager.
* Attempt to access a Leg while logged in as a manager, but accessing a Leg of a Flight that does not belong to them.
* Attempt to retrieve the data of a non-existent Leg.

Test cases 4 – Update:

**Tests.safe**

* Update a Leg with empty data.
* Update a Leg with a variety of erroneous values in its attributes.
* Update a Leg with valid and invalid boundary values for its attributes.
* Update a Leg with a variety of valid values in its attributes.
* Update a Leg to demonstrate the validation that the scheduled departure cannot be in the past.
* Update a Leg to demonstrate the validation that the arrival must be set in the future.
* Update a Leg to demonstrate the validation that the departure and arrival airports cannot be the same.
* Update a Leg to demonstrate the validation that the departure airport does not match the arrival airport of the previous segment.

**Tests.hack**

* Attempt to update a Leg while logged in with a role other than manager.
* Attempt to update a Leg while logged in as a manager, but for a Flight that does not belong to them.
* Attempt to update a Leg while logged in as a manager, but when the Flight is already published.
* Attempt to update a Leg while logged in as a manager, but trying to update it with read-only attributes that should be ignored by the framework.
* Attempt to update a Leg while logged in as a manager, but trying to update it with navigation attributes by providing illegal values.

Test cases 5 – Delete:

**Tests.safe**

* Delete a Leg belonging to the manager.

**Tests.hack**

* Attempt to delete a Leg while logged in with a role other than manager.
* Attempt to delete a Leg while logged in as a manager, but for a Flight that does not belong to them.
* Attempt to delete a Leg while logged in as a manager, but for a published Flight.
* Attempt to delete a Leg while logged in as a manager, but for a Flight that does not exist.

Test cases 6 – Publish:

**Tests.safe**

* Publish a Leg.

**Tests.hack**

* Attempt to publish a Leg while logged in with a role other than manager.
* Attempt to publish a Leg while logged in as a manager, but for a Flight that does not belong to them.
* Attempt to publish a Leg while logged in as a manager, but for a Leg that does not exist.
* Attempt to publish a Leg while logged in as a manager, but for a Leg that is already published.

**Cobertura en base a los servicios de Leg**

A screenshot of a computer

AI-generated content may be incorrect.

In this section dedicated to the test cases for legs, we have achieved extensive coverage with the previously mentioned test cases, reaching an overall final coverage of 99.7% in the leg features. Notably, a code refactoring was undertaken to improve the coverage. This enhancement demonstrates a commitment to code quality and thorough testing, ensuring that most of the code is adequately evaluated and validated.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

The Legs create service doesn't cover that branch because, as the code stands, it's impossible for lastLeg to return null when existingLegs isn't empty. I didn't want to refactor it because we're already in the final days and the validation itself works correctly.

A close-up of a computer code

AI-generated content may be incorrect.

Since there are no tests simulating those three scenarios (a null leg, a mismatched manager, or draft mode being false), the coverage doesn’t exercise all paths, which is why the method is shown in yellow.

A close-up of text

AI-generated content may be incorrect.

You need a test that covers the case where neither departureAirport, arrivalAirport, nor aircraft is selected (all IDs = 0). Since that branch (all three ORs true because they’re zero) isn’t exercised, coverage marks the line in yellow.

## 4.2 Performance Testing

**Introduction to Performance Testing**

In this chapter, we present the performance testing results of our project executed on two different computers. We aim to provide a comprehensive analysis with adequate charts and a 95%-confidence interval for the wall time taken by our project to serve the requests in our functional tests, along with a hypothesis contrast regarding which computer is more powerful.

**Methodology**

We conducted our performance tests on two computers with the following specifications:

Computer A (My laptop):

* Laptop Brand: Asus Rog GL502VS-FY313T
* Operating System: Windows 10 Home
* CPU: Intel® Core™ i7-7700HQ (6M Cache, 2.8GHz hasta 3.8GHz)
* GPU: NVIDIA® GeForce® GTX1070 / 8 GB
* RAM: 32GB DDR4 2400 MHz
* SDD: WD Blue SN580 500GB M.2 PCIe 4.0 NVMe

Computer B (My desktop computer):

* Operating System: Windows 10 Home
* CPU: Intel Core i5-14400F 2.5/4.7GHz
* GPU: GeForce RTX 4060 WINDFORCE OC 8GB GDDR6 DLSS3
* RAM: 32GB DDR4 3600MHz
* SDD: WD Blue SN580 1TB 4150MB/S NVMe PCIe 4.0 M.2 Gen4 16GT/s

Performance results

**Computer A results:**

A graph with blue lines

AI-generated content may be incorrect.

**Computer B results:**

A graph with blue lines

AI-generated content may be incorrect.

**Comparison:**

The comparative analysis of wall times between Computer A and Computer B clearly shows that Computer B is more powerful and efficient in handling the project's requests. Computer B, equipped with a newer generation CPU and potentially benefiting from optimizations in Windows 11, consistently outperforms Computer A.

Computer A shows higher response times for most requests, indicating that it is slower in processing compared to Computer B. Conversely, Computer B demonstrates lower response times, suggesting that it is more efficient and faster. For Computer A, the Most Inefficient Request (MIR) is /airline-manager/leg/publish with the highest response time of approximately 48 ms. For Computer B, the MIR is /airline-manager/leg/publish as well with a response time of approximately 19 ms, which is still lower than the MIR for Computer A.

A screenshot of a computer

AI-generated content may be incorrect.

Based on the image, we can make the following comparisons between the performance of PC A and PC B: The average request time on PC A is significantly higher than on PC B, with PC A averaging 12.54 milliseconds compared to PC B's 5.12 milliseconds, more than twice as long. Additionally, the confidence interval for PC A, with lower and upper bounds of [0.0113, 0.0137], is narrower than that of PC B, which has a confidence interval of [0.0046, 0.0055]. Not only is PC B faster, but its average time is also estimated more accurately.

**Hypothesis contrast**

After performing the analysis of the testing conducted for the test suite, a confidence interval of [0.011302375, 0.013773105] (in seconds) was obtained, which was already deemed acceptable as the upper limit of this interval is much lower than our defined requirement of 1 second. Nevertheless, an improvement in performance has been implemented by adding indices to the tables of the entities due to they help speed some queries up. This way, we can reduce the MIR (Most Inefficient Request). To accomplish this, a hypothesis contrast has been conducted, focusing on analyzing the performance improvement through a z-test and the p-value.

**Preliminary data analysis:**

A screenshot of a computer

AI-generated content may be incorrect.

A graph of blue and black bars

AI-generated content may be incorrect.

At first glance, the performance of flight and leg operations has not improved after adding indexes to the tables: in fact, most endpoints even show a slight deterioration. Our main objective was to speed up the MIR (Most Inefficient Request), /airline-manager/leg/publish, but instead of decreasing, its average response time increased from 48.33 ms to 52.11 ms.

Similarly, the confidence interval for request times has widened, both at its lower and upper limits. The overall average has increased from 11.06 ms to 11.15 ms, but this purely descriptive comparison is not sufficient to conclude whether the change is statistically significant. Therefore, we will perform a hypothesis test (z-test) to confirm, with the appropriate level of significance, whether the observed difference is real or simply a matter of chance.

It's worth noting that, with such a small volume of data (just a few hundred rows), the cost of maintaining and querying indexes is often greater than the benefit they provide: traversing a few hundred records in memory is almost as fast as traversing a B-tree, but without the overhead of managing nodes and indexes. In environments with larger data volumes, indexes do end up amortizing their own cost; with so few rows, the index introduces more latency than it manages to overcome.

**Z-test analysis:**

**Introduction to Z-Test and P-Value**

A z-test is a statistical test used to determine whether there is a significant difference between the means of two samples, which may come from the same or different populations. This test is applicable when the sample sizes are large (typically over 50) and the data is approximately normally distributed.

The p-value is a measure that helps determine the significance of the results in a hypothesis test. It quantifies the evidence against the null hypothesis. The p-value indicates the probability of obtaining test results at least as extreme as the observed results, under the assumption that the null hypothesis is correct.

**Z-Test details**

In our scenario, we conducted a z-test to compare the performance metrics before and after the improvements were made to the system. This test is not paired, meaning it is not necessary for the samples to have the same number of data points.

We set a significance level (alpha, α) to establish the threshold for statistical significance. Commonly, α is set to 0.05, indicating a 5% risk of concluding that a difference exists when there is no actual difference.

**Results of the test**

A screenshot of a computer

AI-generated content may be incorrect.

Computer A Z-test comparing before and after indexes

* P-Value Obtained: 0.91385
* Significance Level (α): 0.05

The p-value obtained from our z-test is 0.91385. To determine the significance of our results, we compare this p-value to our chosen significance level, α.

* If the p-value ≤ α: There is sufficient evidence to reject the null hypothesis in favor of the alternative hypothesis. This indicates a statistically significant difference.
* If the p-value > α: There is not enough evidence to reject the null hypothesis.

In our case, the p-value of 0.91385 is greater than our significance level of 0.05. This result does not fall within the interval [0.00, 0.05), which means:

Conclusion: There is not enough evidence to conclude that the changes (adding indices to the tables of the entities) have had a significant impact on the performance improvement of the system.

**Z-test PC A and PC B analysis:**

**Z-Test details**

In our scenario, we performed a z-test to compare performance metrics between my laptop (PC A) and my desktop (PC B). This test is unpaired, meaning the samples do not need to have the same number of data points.

We set a significance level (alpha, α) to establish the threshold for statistical significance. Typically, α is set at 0.05, indicating a 5% risk of concluding that there is a difference when there actually isn't.

**Results of the test**

A white sheet with black text and numbers

AI-generated content may be incorrect.

Z-test comparing PC A and PC B efficiency

* P-Value Obtained: 0
* Significance Level (α): 0.05

In our case, the p-value of 0 is lower than our significance level of 0.05. This result does fall within the interval [0.00, 0.05), which means:

Conclusion: When comparing the two computers, we found a very large mean difference (z ≈ 10.25) and a p-value that the program rounds to 0 because it is practically zero. Since this p-value is much lower than our significance level of α = 0.05, we reject the hypothesis that the two PCs perform equally. Consequently, we have very strong statistical evidence that PC A and PC B differ in efficiency. This underscores that hardware characteristics and configuration do have a significant influence on performance.

# 5. Conclusions

This report provides a detailed analysis of both the project's functionality and performance.

Functional Testing

A full range of test cases was executed to verify all system features. Most of them worked as expected, and when vulnerabilities emerged, they were immediately corrected, thus strengthening overall security. Test coverage was particularly high in the flight (99.9%) and flight (99.1%) modules. Furthermore, code refactoring efforts contributed to further increasing these percentages, demonstrating my commitment to maintaining a clean and well-tested codebase.

Performance Testing

System response times were compared on two different computers. Computer B proved to be consistently faster than Computer A, reflecting its greater processing power. A statistical comparison confirmed that this difference in efficiency is real and not attributable to chance, highlighting how hardware specifications directly influence performance.

Impact of Indexes on the Database

We evaluated whether adding indexes to tables improved query times. The results, however, did not show a significant improvement, and the z-test found no statistically significant differences.

Conclusion

Overall, the report provides key insights into the robustness of the functionality, operational efficiency, and areas where there is still room for improvement. It also highlights the need to combine extensive testing with a careful analysis of the hardware environment to optimize performance.

# 6. Bibliography

Intentionally blank.