# Testing Report

**Delivery 4**

Acme AirNav Solutions, Inc

Group Number: C1.066
Repository: `https://github.com/mquirosq/Acme-ANS-D04`

Members:
María Quirós Quiroga, marquiqui@alum.us.es
Guillermo Rodríguez Narbona, guirodnar@alum.us.es
Ignacio Mora Pérez, ignmorper1@alum.us.es
Daniel Herrera Urbano, danherurb@alum.us.es
Alejandro Parody Quirós, aleparqui@alum.us.es

May 26, 2025

# Contents

**Executive Summary** **2**

**Revision History** **3**

**1 Introduction** **4**

**2 Functional Testing** **4**

2.1 Administrator/Airport . . . . . 4

2.1.1 List . . . . . 4

2.1.2 Show . . . . . 5

2.1.3 Create . . . . . 5

2.1.4 Update . . . . . 6

2.2 Money Exchange . . . . . 6

2.2.1 Booking . . . . . 6

2.2.2 Flight . . . . . 7

2.2.3 Maintenance Record . . . . . 8

**3 Performance Testing** **9**

3.1 Lenovo Legion . . . . . 9

3.2 MSI Katana . . . . . 10

3.3 Comparison . . . . . 11

**4 Conclusions** **12**

## Executive Summary

This testing report presents the functional test cases recorded for the group features of the Acme AirNav Solutions project, detailing also their usefulness when detecting bugs. The tested features comprise the airport operations implemented for administrator, as well as the integration with a money exchange service. No bugs were detected using these test cases.

Furthermore, this report also presents the average wall times for each request type obtained when executing the aforementioned test suite under two different computers: a Lenovo Legion and an MSI Katana, both of them laptops. Said data is presented in the form of graphs, together with the 95%-confidence interval for both machines.

Finally, both computers were compared using a 95%-confidence hypothesis contrast in order to assess which one delivered the most performance. In this case, the computer with the best performance was the Lenovo Legion.

## Revision History

| Revision | Date | Description |
|---|---|---|
| 1.0 | 2025-05-26 | Initial version |

# 1. Introduction

The aim of this report is to detail the functional test cases recorded for the group features of the Acme AirNav Solutions (Acme ANS for short) project, as well as to analyze the performance delivered by two different computers when executing said test cases.

Therefore, this report will be structured in two chapters. The first one will cover functional testing, presenting all test cases implemented for the airport-related features (listing, showing, creating and updating), as well as for the integration with a money exchange system (for which the exchanges performed for the functionality affected by it were tested). Each test case will also contain a statement concerning its usefulness when finding bugs in the system.

The second chapter, concerning performance testing, will present the data for the average times required by each tested computer to serve the requests present in the test cases, including the 95%-confidence interval, and will also contain a 95%-confidence hypothesis contrast indicating which machine performed better.

# 2. Functional Testing

In this chapter, the test cases implemented for the administrator airport features and the money exchange system integration will be detailed, stating also the extent to which they were useful when finding bugs in the system.

## 2.1. Administrator/Airport

### 2.1.1 List

Safe test cases recorded:

- List all airports, checking all pages available.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to list all airports when signed in as an unauthenticated user and as a user from a different realm, checking that it results in an authorization failure.
  - This test case did not uncover any bugs.

### 2.1.2 Show

Safe test cases recorded:

- Show all airports in the system, checking all screens.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to show an airport as an unauthenticated user and as a user from a different realm, checking that it results in an authentication failure.
  - This test case did not uncover any bugs.

### 2.1.3 Create

Safe test cases recorded:

- Submit the airport creation form with all fields empty, checking that errors are reported.
  - This test case did not uncover any bugs.
- For each field in the airport creation form, leave all other fields empty and set it to each of the positive and negative variations in the `Sample-Data.xlsx` file provided as part of the course materials, using natural intelligence to generate variations for the "phone number" and "IATA code" attributes, checking that the appropriate errors are reported and no exceptions are thrown.
  - This test case did not uncover any bugs.
- Submit the airport creation form with all fields set to valid values, checking that no errors or exceptions are reported, that the airport is correctly created, and that it exists in the system after its creation.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to create an airport while signed in as an unauthenticated user and as a user from a different realm, checking that both result in an authorization failure.
  - This test case did not uncover any bugs.

#### 2.1.4 Update

Safe test cases recorded:

- Submit the airport update form with all fields empty, checking that errors are reported.
  - This test case did not uncover any bugs.
- For each field in the airport update form, leave all other fields empty and set it to each of the positive and negative variations in the `Sample-Data.xlsx` file provided as part of the course materials, using natural intelligence to generate variations for the "phone number" and "IATA code" attributes, checking that the appropriate errors are reported and no exceptions are thrown.
  - This test case did not uncover any bugs.
- Submit the airport update form with all fields set to valid values, checking that no errors or exceptions are reported, that the airport is correctly updated, and that it is shown with modified attributes.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to update an airport with ID 999, with ID xxx, with an empty ID and with no ID parameter, checking that all result in authorization failures.
  - This test case did not uncover any bugs.
- Request to update an airport while signed in as a user from a different realm, checking that it results in an authorization failure.
  - This test case did not uncover any bugs.

### 2.2. Money Exchange

#### 2.2.1 Booking

Safe test cases recorded (tested for both customers and, in the case of show operations, for administrators as well):

- Enter the show screen for a booking with a currency different from the system currency, checking that the exchange is performed.
  - This test case did not uncover any bugs.

- Enter the show screen for a booking with the system currency, checking that no exchange is performed.
  - This test case did not uncover any bugs.
- Change the price of a booking by adding a new passenger, checking that the exchange is computed correctly.
  - This test case did not uncover any bugs.
- Check that the currency is correctly shown when updating a booking, both with and without errors present.
  - This test case did not uncover any bugs.
- Check that the currency is correctly shown when publishing a booking with errors present.
  - This test case did not uncover any bugs.
- Change system currency, checking that booking prices are shown in the new system currency.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Hack the "price" read-only attribute of a booking in the update and publishing forms, ensuring that no exceptions or errors are reported and that no changes are made.
  - This test case did not uncover any bugs.

### 2.2.2 Flight

Safe test cases recorded (for airline managers):

- Enter the show screen for a flight with a currency different from the system currency, checking that the exchange is performed.
  - This test case did not uncover any bugs.
- Enter the show screen for a flight with the system currency, checking that no exchange is performed.
  - This test case did not uncover any bugs.
- Create a new flight with a currency different from the system currency, checking that it is properly created with no errors or exceptions and that an exchange is performed upon showing it.

  - This test case did not uncover any bugs.

- Update flight with erroneous data, checking that the appropriate errors are reported and the currency is properly shown.
  - This test case did not uncover any bugs.
- Update flight without erroneous data, checking that no errors are reported and the currency is properly shown.
  - This test case did not uncover any bugs.
- Publish a flight with erroneous data, checking that the appropriate errors are reported and that the currency is properly shown.
  - This test case did not uncover any bugs.
- Publish a flight with correct data, checking that no errors are reported and that the currency is properly shown.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Hack read-only "system cost" field in the update, publish and delete forms, checking that no errors or exceptions are reported and no changes are made.
  - This test case did not uncover any bugs.
- Request to delete a flight by ID, checking that the currency is correctly displayed in the form.
  - This test case did not uncover any bugs.

### 2.2.3 Maintenance Record

Safe test cases recorded (for technicians):

- Enter the show screen for a maintenance record with a currency different from the system currency, checking that the exchange is performed.
  - This test case did not uncover any bugs.
- Enter the show screen for a maintenance record with the system currency, checking that no exchange is performed.
  - This test case did not uncover any bugs.
- Create a new maintenance record with a different currency, checking that it is correctly created and that an exchange is performed upon showing it.

  - This test case did not uncover any bugs.

- Create a new maintenance record with a different currency, checking that it is correctly created and that an exchange is performed upon showing it.
  - This test case did not uncover any bugs.
- Update a maintenance record, changing its cost, with and without errors present in the form, checking that the exchange performed is different.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Hack the read-only "system cost" attribute of the update form, checking that no exceptions or errors are reported, that no changes are made, and that the currency is correctly displayed.
  - This test case did not uncover any bugs.
- Request to delete a maintenance record by ID, checking that the currency is correctly displayed in the form.
  - This test case did not uncover any bugs.

# 3. Performance Testing

In this chapter, the performance of two different computers when executing the aforementioned test cases will be presented and compared, in order to establish which machine delivered the best performance.

The two computers compared are two laptops belonging to two members of the group:

- Lenovo Legion with an i7 CPU, 16GB of RAM and a 1TB SSD.
- MSI Katana with an i7 CPU, 16GB of RAM and a 500GB SSD.

## 3.1. Lenovo Legion

The system trace file after executing all test cases in this computer was analyzed in Excel following the procedures indicated in the course materials, yielding the following graph for the average wall times:

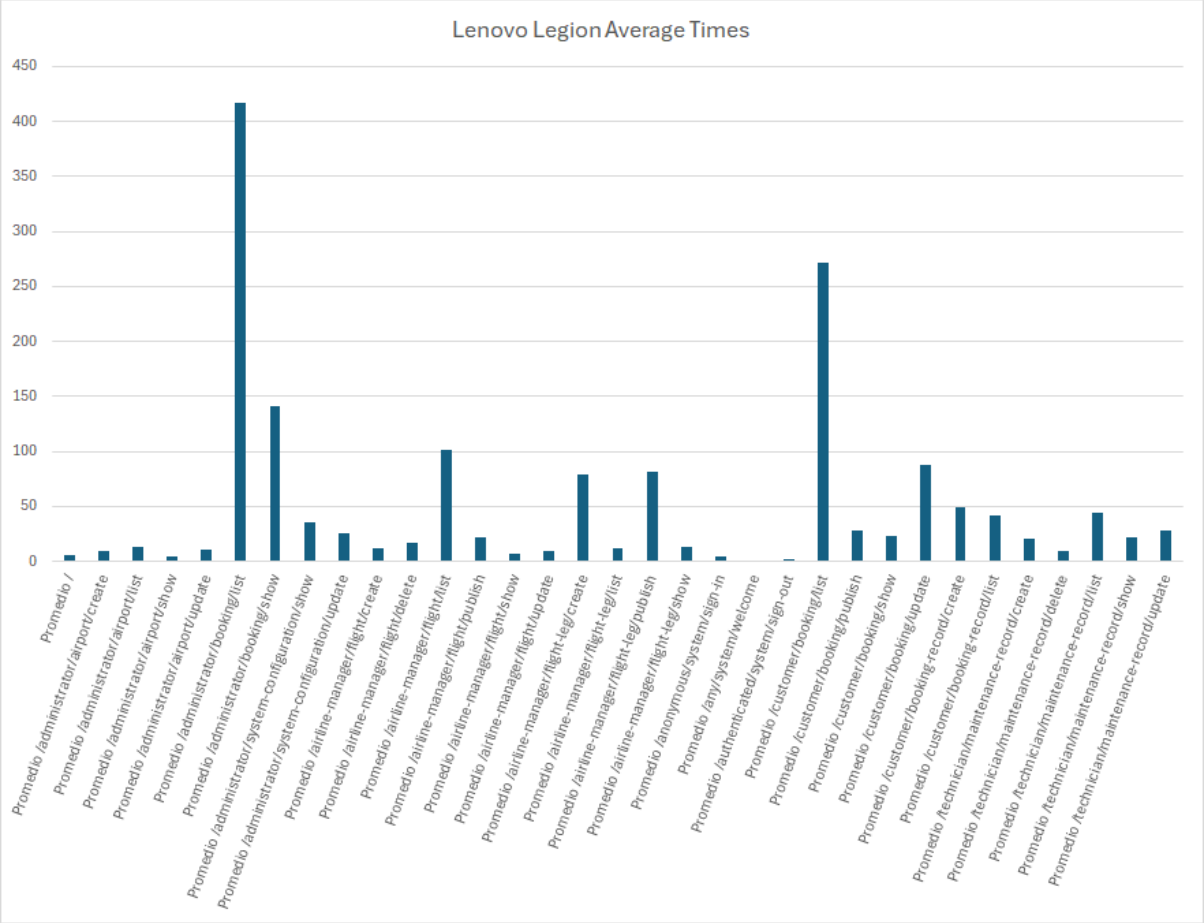


Figure 1: Enter Caption

It can be clearly seen that the MIR (Most Inefficient Request) in this case is the one for the administrator bookings list (`/administrator/booking/list`), which takes an average of 416.7138ms to serve in this computer.

After performing a statistical analysis in Excel following the course indications:

- The average time for the Lenovo Legion is 16.65644419ms.
- The 95%-confidence level is 3.070661706.
- The 95%-confidence interval for the computer will therefore be:
  - [13.58578249, 19.7271059] in milliseconds.
  - [0.013585782, 0.019727106] in seconds.

### 3.2. MSI Katana

The same process was performed for the MSI Katana computer. The average times obtained were the following:

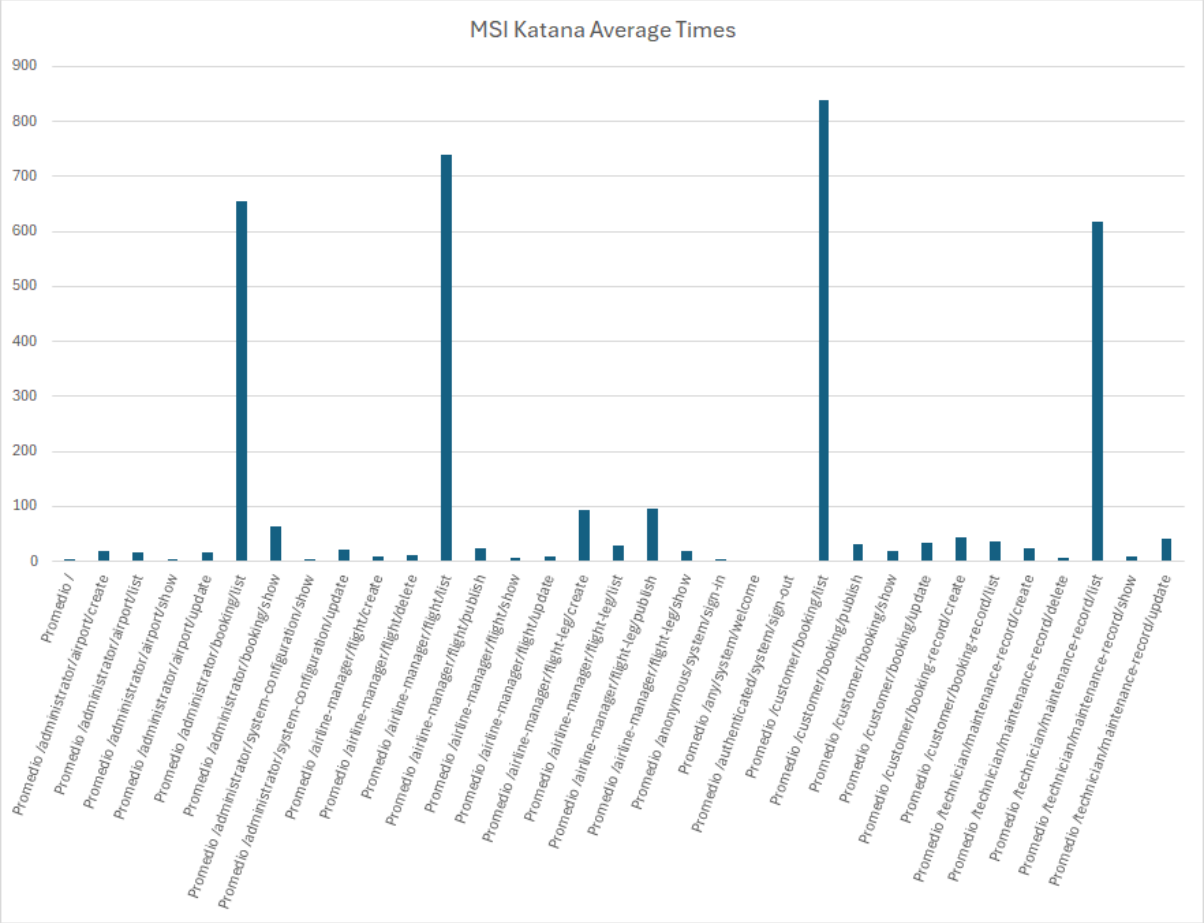


Figure 2: Enter Caption

In the case of this computer, the MIR was the one for the customer bookings list (`/customer/booking/list`), taking an average of 839.6468711ms to serve.

Therefore, for this computer:

- The average time is 38.84898679ms.
- The 95%-confidence level is 8.271028198.
- The 95%-confidence interval for will therefore be:
  - [30,57795859, 47,12001499] in milliseconds.
  - [0,030577959, 0,047120015] in seconds.

### 3.3. Comparison

In order to compare the performance of both computers, we performed a 95%-confidence hypothesis contrast, taking as our initial hypothesis that the Lenovo Legion, which overall has smaller average wall times than the MSI Katana.

In order to verify it, we first have to perform a Z-test, in order to check if the averages can be compared. As the level of confidence was 95%, we used $1 - 0.95 = 0.05$ as the significance level alpha, thus meaning that the P-value for the Z-test must be between 0 and 0.05 in order for the averages to be comparable.

In our case, the Z-test yielded a P-value of 1,49E-06, which is in the aforementioned interval. Therefore, the averages can be compared.

As the average wall time for the Lenovo Legion is of 16.65644419ms and that of the MSI Katana is 38.84898679ms, we can confirm our hypothesis that the Lenovo Legion delivered the most performance during the test execution.

## 4. Conclusions

This report has presented the results of the functional and performance testing processes carried out as part of the Acme ANS project. The functional testing process, despite not uncovering any new bugs in the system, was a valuable tool for validating the design choices made when implementing the features of the project, as well as for ensuring that said features are secure.

The performance analysis reveals the most inefficient requests in the set of features tested to be those associated with list operations, which involve fetching large amounts of data from the database. The performance comparison between the Lenovo Legion and the MSI Katana (in which the Lenovo Legion delivered the best performance) gave us an insight into the differences to be expected when executing the system under different machines and different conditions.

## References

Intentionally blank