Testing Report

Acme AirNav Solutions, Inc

Group Number: C1.066
Repository: `https://github.com/mquirosq/DP2-C1.066`

Student #1:
Ignacio Mora Pérez, ignmorper1@alum.us.es

May 26, 2025

Contents

Executive Summary

In this document, the testing process carried out during the fourth Acme-ANS deliverable will be described. Specifically, those tests cases regarding the functionalities implemented by Student #1. The goal of this process is to ensure the security of the project, trying to expect any possible hacking attempt.

Different test cases performed over each feature will be described in detail in the functional testing section. Those cases cover the functionalities of airline managers over flight and flight legs. They can be classified in two types: Safe and hacking test cases. The former corresponds to those actions provided by the application (legal), ensuring that they are correctly limited. The latter have been carried out by trying to access features that are not shown in the user interface, but that could be tried to be accessed via URL (illegal).

The last part of the report covers performance testing. The efficiency of the application has been measured, using two different devices. This will help us to determine if any issues identified are caused by hardware components, or by the complexity of the application's software. Statistical metrics are provided for the performance of both devices.

Revision History

Revision	Date	Description
1.0	2025-05-26	Initial version

1. Introduction

This document will describe the results of the recording of test cases performed over the features implemented by Student #1. The structure of the document is described below.

The first section, includes information relative to functional testing. A listing with the test cases implemented, grouped by feature will be provided. For each test case, a succinct description plus an indication of how effective it was at detecting bugs is included.

The second section includes information relative to performance testing. Charts showing this performance, as well as a 95%-confidence interval for the wall time taken by the project to serve the requests of the functional tests in two different computers. Finally, a 95%-confidence hypothesis compares the performance of both computers to conclude which is the most powerful.

2. Functional testing

This section consists of a list of the tested cases implemented , grouped by feature.

2.1. Operations of airline managers on flights

2.1.1 List

The safe cases implemented include:

- For each airline manager, display the list of their flights.
  - Detection of bugs: No bugs were detected.

The hacking cases implemented include:

- Attempt to display the list of flights as an unauthenticated user, and as a customer. Both cases must throw an authorization exception.
  - Detection of bugs: No bugs were detected.

2.1.2 Show

The safe cases implemented include:

- The show feature has been requested for each of the flights in the sample data.
  - Detection of bugs: No bugs were detected.

The hacking cases implemented include:

- The show feature was requested, but invalid values were introduced in the ID field. The values introduced were: "XXX", -1, 999, "" (empty), and without the ID parameter. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The show feature is requested as an unauthenticated user, and as a customer. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The show feature is requested by manager2, for a flight belonging to manager1. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.

2.1.3 Create

The safe cases implemented include:

- Submit an empty form to check that no exceptions were thrown and errors were reported in the correct attributes.
  - Detection of bugs: No bugs were detected.
- For each of the attributes in the creation form, invalid data variations are introduced. As no particular cases have been identified, this data has been introduced with the help of the "Sample-Data" file. All submissions must display errors on the attribute tested.
  - Detection of bugs: No bugs were detected.
- Submit a form with valid data to check that the flight is created properly.
  - Detection of bugs: No bugs were detected.

The hacking cases implemented include:

- As for the previous features, the creation of flights has tried to be accessed as an unauthenticated user, and as a customer. Both requests must result in an authorization exception.
  - Detection of bugs: No bugs were detected.

2.1.4 Update

The safe cases implemented include:

- Submit an empty form to check that no exceptions were thrown and errors were reported in the correct attributes.
  - Detection of bugs: No bugs were detected.
- Similarly as in the previous feature, different invalid data variations have been introduced in each of the attributes of the displayed form. In this case, as the information of the flight being edited is present, all data must be erased before starting. The "Sample-Data" file has been used as well. All submissions must display errors on the attribute tested.
  - Detection of bugs: No bugs were detected.
- Submit a form with valid data to check that the flight is properly updated.
  - Detection of bugs: No bugs were detected.

The hacking cases implemented include:

- The update feature of a flight has tried to be accessed as an unauthenticated user, and as a customer. Both requests must result in an authorization exception.
  - Detection of bugs: No bugs were detected.
- The update feature is requested by manager1, for a flight belonging to manager2. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The update feature is requested for a published flight. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The update feature is requested, but invalid values are introduced in the ID field. The values introduced are: "XXX", -1, 999, "" (empty), and without the ID parameter. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- In the displayed form, all read-only attributes of a flight are tried to be modified. Even though an authorization exception is not thrown, the values introduced must be ignored. This was done with the help of the Developer Tools found on the browser.
  - Detection of bugs: No bugs were detected.

2.1.5 Publish

The safe cases implemented include:

- Submit an empty form to check that no exceptions were thrown and errors were reported in the correct attributes.
  - Detection of bugs: No bugs were detected.
- Invalid data variations have been introduced in each of the attributes of the displayed form. In this case, as the information of the flight being edited is present, all data must be erased before starting. The "Sample-Data" file has been used as well. All submissions must display errors on the attribute tested.
  - Detection of bugs: No bugs were detected.
- Submit a form with valid values, for a flight that does not have any leg. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- Submit a form with valid values, for a flight that has unpublished legs. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- Submit a form with valid data to check that the flight is properly published.
  - Detection of bugs: No bugs were detected.

The hacking cases implemented include:

- The publish feature of a flight has tried to be accessed as an unauthenticated user, and as a customer. Both requests must result in an authorization exception.
  - Detection of bugs: No bugs were detected.
- The publish feature is requested by manager1, for a flight belonging to manager2. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The publish feature is requested for an already published flight. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The publish feature is requested, but invalid values are introduced in the ID field. The values introduced are: "XXX", -1, 999, "" (empty), and without the ID parameter. An authorization exception must be thrown.

  - Detection of bugs: No bugs were detected.

- In the displayed form, all read-only attributes of a flight are tried to be modified. Even though an authorization exception is not thrown, the values introduced must be ignored. This was done with the help of the Developer Tools found on the browser.

  - Detection of bugs: No bugs were detected.

2.1.6 Delete

The safe cases implemented include:

- A flight without legs is tried to be deleted. The flight must be properly deleted.

  - Detection of bugs: No bugs were detected.

- A flight with only unpublished legs is tried to be deleted. The flight must be properly deleted.

  - Detection of bugs: No bugs were detected.

- A flight with published legs is tried to be deleted. An error must be displayed.

  - Detection of bugs: No bugs were detected.

- A flight with published legs is tried to be deleted. The currency of the cost of the flight must be different that the system currency. An error must be displayed, and the cost in the system currency must be correctly computed and displayed.

  - Detection of bugs: No bugs were detected.

The hacking cases implemented include:

- The delete feature of a flight has tried to be accessed as an unauthenticated user, and as a customer. Both requests must result in an authorization exception.

  - Detection of bugs: No bugs were detected.

- The delete feature is requested by manager1, for a flight belonging to manager2. An authorization exception must be thrown.

  - Detection of bugs: No bugs were detected.

- The delete feature is requested for an already published flight. An authorization exception must be thrown.

  - Detection of bugs: No bugs were detected.

- The delete feature is requested, but invalid values are introduced in the ID field. The values introduced are: "XXX", -1, 999, "" (empty), and without the ID parameter. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.

2.2. Operations of airline managers on flight legs

2.2.1 List

The safe cases implemented include:

- Display the list of flight legs of each of the flights of each of the airline managers.
  - Detection of bugs: No bugs were detected.

The hacking cases implemented include:

- The list feature of the flight legs of a flight has tried to be accessed as an unauthenticated user, and as a customer. Both requests must result in an authorization exception.
  - Detection of bugs: No bugs were detected.
- The list feature is requested by manager1, for the flight legs of a flight belonging to manager2. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The list feature is requested, but invalid values are introduced in the flight's ID field. The values introduced are: "XXX", -1, 999, "" (empty), and without the ID parameter. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.

2.2.2 Show

The safe cases implemented include:

- The show feature has been requested for each of the legs in the sample data.
  - Detection of bugs: No bugs were detected.

The hacking cases implemented include:

- The show feature of a flight leg has tried to be accessed as an unauthenticated user, and as a customer. Both requests must result in an authorization exception.
  - Detection of bugs: No bugs were detected.
- The show feature is requested by manager1, for the flight legs belonging to manager1. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The show feature is requested, but invalid values are introduced in the flight leg's ID field. The values introduced are: "XXX", -1, 999, "" (empty), and without the ID parameter. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.

2.2.3 Create

The safe cases implemented include:

- Submit an empty form to check that no exceptions were thrown and errors were reported in the correct attributes.
  - Detection of bugs: No bugs were detected.
- For each of the attributes in the creation form, invalid data variations are introduced. All attributes in this form presented particular cases:
  - For the flightNumber attribute, different pattern variations have been introduced to ensure that the first part of the pattern does not accept lower case letters, numbers, special characters, or a number of characters different from 3. This part of the pattern must also correspond to the airline's IATA code, what was also tested. For the second part of the pattern, values have been introduced to check that letters, special characters, or a number of digits different from 4 are not accepted. Then, the uniqueness of the attribute has also been tested by introducing already existing flightNumber values.
  - Scheduled departure and arrival values must be in the future. Different combinations of these two are introduced to ensure that the scheduled departure is before (and not equal to) the scheduled arrival. The upper limit of both attributes is also tested (determined by the framework).
  - All the possible values for the status attribute have been introduced. All these values must be accepted.
  - For the departure and arrival airports, the same value is introduced to ensure that they can not coincide. This must result in an error.
  - Detection of bugs: Even though the restriction for the scheduled departure to be before the arrival was correctly implemented, there was no error shown when introducing an empty scheduled departure and a scheduled arrival in

the past. The restriction for the scheduled arrival to be in the future was implemented to solve this.

- Submit a form with valid data to check that the flight leg is properly created.
  - Detection of bugs: No bugs were detected.

The hacking cases implemented include:

- The create feature of a flight leg has tried to be accessed as an unauthenticated user, and as a customer. Both requests must result in an authorization exception.
  - Detection of bugs: No bugs were detected.
- The create feature is requested by manager1, to create a flight leg for a flight belonging to manager2. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The create feature is requested to create a flight leg for an already published flight. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The create feature is requested, but invalid values are introduced in the flight's ID field. The values introduced are: "XXX", -1, 999, "" (empty), and without the ID parameter. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.

2.2.4 Update

The update feature follows a similar approach as the create feature. The safe cases implemented include:

- Submit an empty form to check that no exceptions were thrown and errors were reported in the correct attributes.
  - Detection of bugs: No bugs were detected.
- For each of the attributes in the update form, invalid data variations are introduced. All attributes in this form presented particular cases:
  - All the restrictions that were tested in the creation of flight legs are repeated for the flightNumber. In addition, the flight leg's previous flightNumber must be introduced, in order to check that it is not counted as a repeated value.
  - All the restrictions that were tested in the creation of flight legs are repeated for the scheduled departure and arrival attributes.

  - All the restrictions that were tested in the creation of flight legs are repeated for the status attribute.
  - All the restrictions that were tested in the creation of flight legs are repeated for the departure and arrival airports attributes.
  - Detection of bugs: No bugs were detected.

- Submit a form with valid data to check that the flight leg is properly updated.
  - Detection of bugs: No bugs were detected.

The hacking cases implemented include:

- The update feature of a flight leg has tried to be accessed as an unauthenticated user, and as a customer. Both requests must result in an authorization exception.
  - Detection of bugs: No bugs were detected.
- The update feature is requested by manager2, to update a flight leg belonging to manager1. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The update feature is requested to update a published flight leg. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The update feature is requested, but invalid values are introduced in the flight leg's ID field. The values introduced are: "XXX", -1, 999, "" (empty), and without the ID parameter. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- In the displayed form, all read-only attributes of a flight leg are tried to be modified. Even though an authorization exception is not thrown, the values introduced must be ignored. This was done with the help of the Developer Tools found on the browser.
  - Detection of bugs: No bugs were detected.

2.2.5 Publish

The publish feature follows a similar approach as both the update and the create feature. The safe cases implemented include:

- Submit an empty form to check that no exceptions were thrown and errors were reported in the correct attributes.

  - Detection of bugs: No bugs were detected.

- For each of the attributes in the publish form, invalid data variations are introduced. All attributes in this form presented particular cases:
  - All the restrictions that were tested in the update feature of flight legs are repeated for the flightNumber.
  - All the restrictions that were tested in the update feature of flight legs are repeated for the scheduled departure and arrival attributes. In addition, cases in which the manager tries to publish a flight leg which overlaps with other published flight legs of the same flight are tested. All the combinations in which the leg being published overlaps with other published legs must result in an error. In any other combination, no error must be displayed.
  - All the restrictions that were tested in the update feature of flight legs are repeated for the status attribute.
  - All the restrictions that were tested in the update feature of flight legs are repeated for the departure and arrival airports attributes.
  - Detection of bugs: No bugs were detected.

- Submit a form with valid data to check that the flight leg is properly published.
  - Detection of bugs: No bugs were detected.

The hacking cases implemented include:

- The publish feature of a flight leg has tried to be accessed as an unauthenticated user, and as a customer. Both requests must result in an authorization exception.
  - Detection of bugs: No bugs were detected.

- The publish feature is requested by manager2, to publish a flight leg belonging to manager1. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.

- The publish feature is requested to publish an already published flight leg. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.

- The publish feature is requested, but invalid values are introduced in the flight leg's ID field. The values introduced are: "XXX", -1, 999, "" (empty), and without the ID parameter. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.

- In the displayed form, all read-only attributes of a flight leg are tried to be modified. Even though an authorization exception is not thrown, the values introduced must be ignored. This was done with the help of the Developer Tools found on the browser.
  - Detection of bugs: No bugs were detected.

2.2.6 Delete

The safe cases implemented include:

- A flight leg in draft mode is tried to be deleted. The flight leg must be properly deleted.
  - Detection of bugs: There was unnecessary code in the unbind method of the delete feature, that was never executed. This code was removed.

The hacking cases implemented include:

- The delete feature of a flight leg has tried to be accessed as an unauthenticated user, and as a customer. Both requests must result in an authorization exception.
  - Detection of bugs: No bugs were detected.
- The delete feature is requested by manager2, for a flight leg belonging to manager1. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The delete feature is requested for an already published flight leg. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.
- The delete feature is requested, but invalid values are introduced in the ID field. The values introduced are: "XXX", -1, 999, "" (empty), and without the ID parameter. An authorization exception must be thrown.
  - Detection of bugs: No bugs were detected.

3. Performance testing

In this section, we will evaluate the performance of the project by measuring wall time, the elapsed real time taken to complete requests during functional tests. The goal is to assess how quickly the system responds under real conditions to determine which of two computers performs better.

To collect the data required to compute the mean confidence interval and perform the hypothesis contrast, we will run all functional tests for customer features. We will be using the following devices:

- Macbook Air M1 (Virtual machine): 8GB RAM.
- Windows Laptop: 16GB RAM.

3.1. Mean confidence interval

After treating the data obtained from the *"tester.trace"* file, I have performed an analysis using Excel and following the proceedings in the slides to obtain the average wall time for each of the requests executed in the tests.

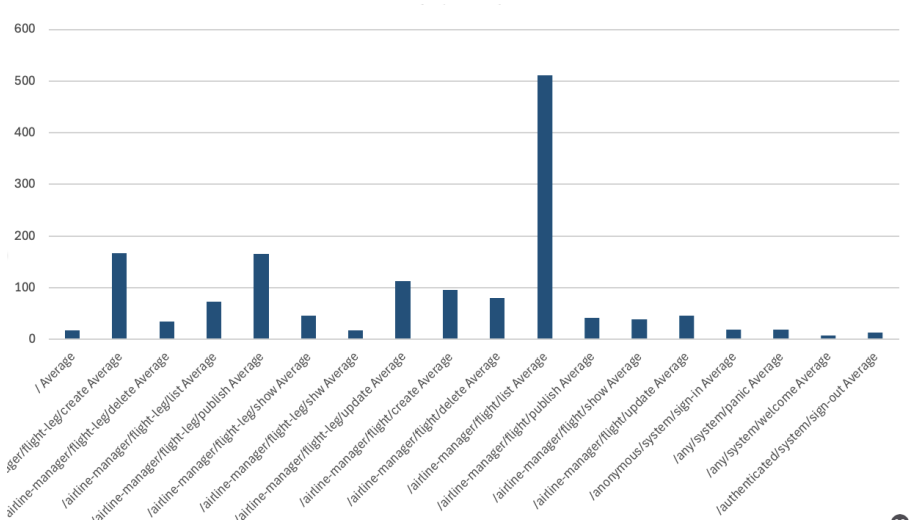

Figure 1: Graph showing average execution time for each request in the Macbook Air

For the Macbook Air, we obtained a grand average of 75,67 ms. As we can see in the graph1, the MIR (Most Inefficient Request) is the flight listing, whose average stands at 511,14 ms, much higher than the rest of the requests.

Using the Excel data analyzer, we obtain the amplitude of the confidence interval at 95%, 8,36 ms. By removing this amount from the average and adding it, we obtain the confidence interval: [67,3 ms - 84,04 ms] or [0,0673 s - 0,08404 s].

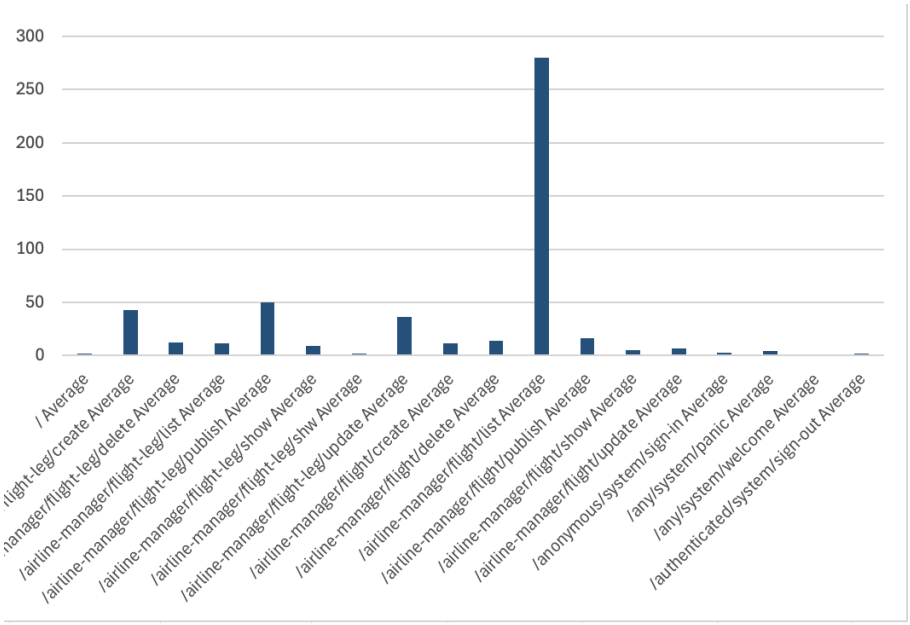

Figure 2: Graph showing average execution time for each request in the Windows PC

On the other hand, when doing the same with the Windows PC we obtained a graph 2 with a similar shape. In this case, the grand average stands at 25,35 ms, much lower than for the first laptop. Once again, the MIR is the flight listing, with an average of 279,36 ms.

Using the Excel data analyzer, we obtain the amplitude of the confidence interval at 95%, 3,73 ms. By removing this amount from the average and adding it, we obtain the confidence interval: [21,61 ms - 29,08 ms] or [0,02161 s - 0,02908 s].

4. Conclusions

This document serves as a means of reporting the test cases performed for future reference. The analysis recorded in this document provides a detailed evaluation of the test cases in order to identify possible bugs in the code, evaluate potential solutions, and make informed decisions to improve the quality of the code base and ensure protection against security breaches.

Each test case was recorded in a structured manner, grouped by feature, and reporting on the bugs that were detected thanks to this test case. This report serves as a transparent reference for the testing process.

The testing report provides a clear explanation for the tests performed. In the future, these tests will serve as a solid foundation for continuous improvement and quality assurance.

References

[1] *Requirements – Student #1*, Enseñanza Virtual, 2025. Available on Enseñanza Virtual in the course inside the Project Statement folder.