Testing Report

Guillermo Rodríguez Narbona (Student #4)

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

In this testing report, the functional test cases recorded for the student #4 features of the Acme AirNav Solutions (Acme ANS for short) project will be detailed, including their usefulness in bug detection. The tested features were the listing, showing, creation, updating, publishing and deletion of both claims and tracking logs. Two bugs were detected in the project thanks to these test cases, and were both solved.

Furthermore, the performance of the test suite composed of all the aforementioned test cases was compared across two different computers, a Lenovo Legion laptop and a HP Victus laptop. The 95%-confidence interval for the wall time taken to serve each type of request was also computed for both computers, and a 95%-confidence hypothesis contrast between the data for both machines revealed the Lenovo Legion to be the one that delivered the most performance.

Revision History

Revision	Date	Description
1.0	2025-05-25	Initial version
1.1	2025-05-26	Added executive summary, introduction and conclusion

1. Introduction

The aim of this testing report is to discuss the functional and performance testing processes carried out for all the functionality implemented as part of the student #4 requirements (listing, showing, creating, updating, publishing and deleting of claims and tracking logs). The functional testing process had as a goal to detect any potential bugs in the system that had not yet been found via informal testing and to ensure that every tested feature had proper security, while the performance testing process focused on comparing the performance between two computers running the system.

Therefore, this testing report is organized into two chapters. First, a chapter on functional testing, in which all test cases implemented for the student #4 functionality will be detailed, including the bugs each test case helped detect. Afterwards, the performance testing chapter will discuss the performance of the system under two different computers when running the aforementioned test cases, comparing their performance in order to assess which one was more powerful.

2. Functional Testing

In this chapter, the test cases implemented for the student #4 features will be detailed, grouped by the feature to which they belong, stating also the extent to which they were useful when finding bugs in the system.

2.1. Assistance Agent/Claim

2.1.1 List

Safe test cases recorded:

- List all claims associated to `agent1`, `agent2` and `agent3`, checking all available pages.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- List claims as an unauthenticated user and as a user from a different realm, checking that both result in an authorization failures.
  - This test case did not uncover any bugs.

2.1.2 Show

Safe test cases recorded:

- Show all claims associated to `agent1`, `agent2` and `agent3`, accessing each of the screens.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to show a claim belonging to `agent1` as an unauthenticated user, as a user from a different realm, and as a different user with the same realm (`agent2`), checking that all result in an authorization failure.
  - This test case did not uncover any bugs.
- Request to show a claim with ID -1, with ID `xxx`, with an empty ID, and with no ID parameter, to check that they all result in an authorization failure.
  - This test case uncovered a bug in the authorization for the claim show, update, publish and delete features, as the case in which no ID parameter was present was not being handled by the corresponding services.

2.1.3 Create

Safe test cases recorded:

- Submit the claim creation form with all fields empty, checking that errors are reported and no exceptions occur.
  - This test case did not uncover any bugs.
- Submit the claim creation form with all fields empty but one, trying all positive and negative variations of said field provided in the `Sample-Data.xlsx` file, checking that no exceptions occur and errors are reported. This was done for all fields present in the form.
  - This test case did not uncover any bugs.
- Submit the claim creation form with all fields set to valid attributes, checking that a claim is properly created, without errors or exceptions, 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 a claim 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.
- Hack the `status` read-only field of the creation form, and set it to both an empty value and a value different from the one initially shown, checking that no errors or exceptions occur and no changes are made.
  - This test case did not uncover any bugs.
- Hack the `registrationMoment` read-only field of the creation form, and set it to both an empty value and a value different from the one initially shown, checking that no errors or exceptions occur and no changes are made.
  - This test case did not uncover any bugs.
- Hack the `leg` navigation attribute of the creation form, and set it to an empty ID, ID `xxx`, an unpublished leg ID and the ID of a leg with arrival date in the future, checking that it results in an authorization error.
  - This test case did not uncover any bugs.

2.1.4 Update

Safe test cases recorded:

- Submit the claim update form with all fields empty, checking that errors are reported and no exceptions occur.
  - This test case did not uncover any bugs.
- Submit the claim update form with all fields empty but one, trying all positive and negative variations of said field provided in the `Sample-Data.xlsx` file, checking that no exceptions occur and errors are reported. This was done for all fields present in the form.
  - This test case helped uncover a bug by which flight legs to which to associate the updated claim had their arrival dates compared with respect to the current moment as opposed to the claim's creation moment.
- Submit the claim update form with all fields set to valid attributes, checking that the claim is properly updated, without errors or exceptions, and that it is shown with its updated values.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to update a claim belonging to `agent1` as an unauthenticated user, as a user from a different realm, and as a different user with the same realm (`agent2`), checking that all result in an authorization failure.
  - This test case did not uncover any bugs.
- Request to update a published claim, checking that it results in an authorization failure.
  - This test case did not uncover any bugs.
- Request to update a claim with ID -1, with ID `xxx`, with an empty ID, and with no ID parameter, to check that they all result in an authorization failure.
  - This test case did not uncover any bugs.
- Hack the `status` read-only field of the update form, and set it to both an empty value and a value different from the one initially shown, checking that no errors or exceptions occur and no changes are made.
  - This test case did not uncover any bugs.
- Hack the `registrationMoment` read-only field of the update form, and set it to both an empty value and a value different from the one initially shown, checking that no errors or exceptions occur and no changes are made.
  - This test case did not uncover any bugs.
- Hack the `leg` navigation attribute of the update form, and set it to an empty ID, ID `xxx`, an unpublished leg ID and the ID of a leg with arrival date in the future, checking that it results in an authorization error.
  - This test case did not uncover any bugs.

2.1.5 Publish

Safe test cases recorded:

- Submit the claim publishing form with all fields empty, checking that errors are reported and no exceptions occur.
  - This test case did not uncover any bugs.
- Submit the claim publishing form with all fields empty but one, trying all positive and negative variations of said field provided in the `Sample-Data.xlsx` file, checking that no exceptions occur and errors are reported. This was done for all fields present in the form.

  - This test case did not uncover any bugs.

- Submit the claim publishing form with all fields set to valid attributes, checking that the claim is properly published, without errors or exceptions, and that it is shown as published.

  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to publish a claim belonging to `agent1` as an unauthenticated user, as a user from a different realm, and as a different user with the same realm (`agent2`), checking that all result in an authorization failure.

  - This test case did not uncover any bugs.

- Request to publish an already published claim and an uncompleted claim, checking that both result in an authorization failure.

  - This test case did not uncover any bugs.

- Request to publish a claim with ID -1, with ID `xxx`, with an empty ID, and with no ID parameter, to check that they all result in an authorization failure.

  - This test case did not uncover any bugs.

- Hack the `status` read-only field of the publishing form, and set it to both an empty value and a value different from the one initially shown, checking that no errors or exceptions occur and no changes are made.

  - This test case did not uncover any bugs.

- Hack the `registrationMoment` read-only field of the publishing form, and set it to both an empty value and a value different from the one initially shown, checking that no errors or exceptions occur and no changes are made.

  - This test case did not uncover any bugs.

- Hack the `leg` navigation attribute of the publishing form, and set it to an empty ID, ID `xxx`, an unpublished leg ID and the ID of a leg with arrival date in the future, checking that it results in an authorization error.

  - This test case did not uncover any bugs.

2.1.6 Delete

Safe test cases recorded:

- Delete a claim, checking that no errors or exceptions appear and that it no longer exists in the listing.

- This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to delete a claim belonging to `agent1` as an unauthenticated user, as a user from a different realm, and as a different user with the same realm (`agent2`), checking that all result in an authorization failure.
  - This test case did not uncover any bugs.
- Request to delete a published claim, checking that it results in an authorization failure.
  - This test case did not uncover any bugs.
- Request to delete a claim with ID -1, with ID `xxx`, with an empty ID, and with no ID parameter, to check that they all result in an authorization failure.
  - This test case did not uncover any bugs.
- Hack the `leg` navigation attribute of the deletion form, and set it to an empty ID, ID `xxx`, an unpublished leg ID and the ID of a leg with arrival date in the future, checking that it results in an authorization error.
  - This test case did not uncover any bugs.

2.2. Assistance Agent/Tracking Log

2.2.1 List

Safe test cases recorded:

- List all tracking logs associated to all claims of `agent1`, `agent2` and `agent3`, checking all available pages.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to list the tracking logs associated to a claim belonging to `agent1` as an unauthenticated user, as a user from a different realm, and as a different user with the same realm (`agent2`), checking that all result in an authorization failure.
  - This test case did not uncover any bugs.

- Request to list tracking logs associated to a claim with ID -1, with ID `xxx`, with an empty ID, and with no ID parameter, to check that they all result in an authorization failure.
  - This test case did not uncover any bugs.

2.2.2 Show

Safe test cases recorded:

- Show all tracking logs associated to all claims of `agent1`, `agent2` and `agent3`, accessing each screen.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to show a tracking log belonging to `agent1` as an unauthenticated user, as a user from a different realm, and as a different user with the same realm (`agent2`), checking that all result in an authorization failure.
  - This test case did not uncover any bugs.
- Request to show a tracking log with ID -1, with ID `xxx`, with an empty ID, and with no ID parameter, to check that they all result in an authorization failure.
  - This test case did not uncover any bugs.

2.2.3 Create

Safe test cases recorded:

- Submit the tracking log creation form with all fields empty, checking that errors are reported and no exceptions occur.
  - This test case did not uncover any bugs.
- Submit the tracking log creation form with all fields empty but one, trying all positive and negative variations of said field provided in the `Sample-Data.xlsx` file, checking that no exceptions occur and errors are reported. This was done for all fields present in the form.
  - This test case did not uncover any bugs.

- Submit the tracking log creation form with a resolution percentage lower than 100% and in `ACCEPTED` and `REJECTED` status, as well as with a resolution percentage of 100% and `PENDING` status, in order to check that errors are reported.
  - This test case did not uncover any bugs.
- Submit the tracking log creation form with `ACCEPTED` and `REJECTED` status and no value in the "resolution" field, in order to check that errors are reported, as it is mandatory in case the status is not `PENDING`.
  - This test case did not uncover any bugs.
- Submit the tracking log creation form with all fields set to valid attributes for a `PENDING` tracking log, checking that a tracking log is properly created, without errors or exceptions, and that it exists in the system after its creation.
  - This test case did not uncover any bugs.
- Submit the tracking log creation form with a resolution percentage lower than that of the previously created tracking log to check that errors are reported, as the resolution percentage must be monotonically increasing.
  - This test case did not uncover any bugs.
- Submit the tracking log creation form with a resolution percentage of 100% (higher than the previous one), checking that a completed tracking log is correctly created without errors or exceptions and it exists in the system.
  - This test case did not uncover any bugs.
- Submit the tracking log creation form with a resolution percentage lower than the previous one, checking that errors are reported.
  - This test case did not uncover any bugs.
- Submit the tracking log creation form to create an exceptional ("reclaimed") tracking log without having a previously completed tracking log already published, checking that the appropriate error is reported.
  - This test case did not uncover any bugs.
- Submit the tracking log creation form to create an exceptional ("reclaimed") tracking log having a previously completed tracking log already published, with a status different from the one of the previously completed tracking log, checking that the appropriate error is reported.
  - This test case did not uncover any bugs.
- Submit the tracking log creation form with correct attributes to create an exceptional ("reclaimed") tracking log, checking that an exceptional tracking log is created without errors or exceptions and that it exists in the system.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to create a tracking log for a claim belonging to `agent1` as an unauthenticated user, as a user with a different realm, and as a different user with the same realm, checking that all result in an authorization failure.
  - This test case did not uncover any bugs.
- Request to create a tracking log for a claim with ID -1, with ID `xxx`, with an empty ID, with no ID parameter, and for a claim for which no more tracking logs can be created (as it is completed and has an exceptional tracking log), to check that they all result in an authorization failure.
  - This test case did not uncover any bugs.
- Hack the `creationMoment` and `lastUpdateMoment` read-only fields of the creation form, and set them to both empty values and values different from the ones initially shown, checking that no errors or exceptions occur and no changes are made.
  - This test case did not uncover any bugs.

2.2.4 Update

Safe test cases recorded:

- Submit the tracking log update form with all fields empty, checking that errors are reported and no exceptions occur.
  - This test case did not uncover any bugs.
- Submit the tracking log update form with all fields empty but one, trying all positive and negative variations of said field provided in the `Sample-Data.xlsx` file, checking that no exceptions occur and errors are reported. This was done for all fields present in the form.
  - This test case did not uncover any bugs.
- Submit the tracking log update form with a resolution percentage lower than 100% and in `ACCEPTED` and `REJECTED` status, as well as with a resolution percentage of 100% and `PENDING` status, in order to check that errors are reported.
  - This test case did not uncover any bugs.
- Submit the tracking log update form with `ACCEPTED` and `REJECTED` status and no value in the "resolution" field, in order to check that errors are reported, as it is mandatory in case the status is not `PENDING`.
  - This test case did not uncover any bugs.

- Update a `PENDING` tracking log with correct attributes, checking that that the tracking log is properly updated, without errors or exceptions, and that it exists in the system with its modified attributes.
  - This test case did not uncover any bugs.
- Update a tracking log after the previous one with a higher resolution percentage, checking that no errors or exceptions are reported and the tracking log has its attributes modified.
  - This test case did not uncover any bugs.
- Update an exceptional ("reclaimed") tracking log with a resolution percentage lower than 100%, checking the appropriate error is reported.
  - This test case did not uncover any bugs.
- Update an exceptional ("reclaimed") tracking log, with a status different from the one of the previously completed tracking log, checking that the appropriate error is reported.
  - This test case did not uncover any bugs.
- Submit the tracking log update form with correct attributes to update an exceptional ("reclaimed") tracking log, checking that an exceptional tracking log is updated without errors or exceptions and that it exists in the system with its modified attributes.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to update a tracking log belonging to `agent1` as an unauthenticated user, as a user with a different realm, and as a different user with the same realm, checking that all result in an authorization failure.
  - This test case did not uncover any bugs.
- Request to update a tracking log with ID -1, with ID `xxx`, with an empty ID, with no ID parameter, and with the ID of an already published tracking log, to check that they all result in an authorization failure.
  - This test case did not uncover any bugs.
- Hack the `creationMoment` and `lastUpdateMoment` read-only fields of the update form, and set them to both empty values and values different from the ones initially shown, checking that no errors or exceptions occur and no changes are made.
  - This test case did not uncover any bugs.

2.2.5 Publish

Safe test cases recorded:

- Submit the tracking log publishing form with all fields empty, checking that errors are reported and no exceptions occur.
  - This test case did not uncover any bugs.
- Submit the tracking log publishing form with all fields empty but one, trying all positive and negative variations of said field provided in the `Sample-Data.xlsx` file, checking that no exceptions occur and errors are reported. This was done for all fields present in the form.
  - This test case did not uncover any bugs.
- Submit the tracking log publishing form with a resolution percentage lower than 100% and in `ACCEPTED` and `REJECTED` status, as well as with a resolution percentage of 100% and `PENDING` status, in order to check that errors are reported.
  - This test case did not uncover any bugs.
- Submit the tracking log publishing form with `ACCEPTED` and `REJECTED` status and no value in the "resolution" field, in order to check that errors are reported, as it is mandatory in case the status is not `PENDING`.
  - This test case did not uncover any bugs.
- Submit the tracking log publishing form with all fields set to valid attributes for a `PENDING` tracking log, checking that a tracking log is properly published, without errors or exceptions, and that it is displayed as published.
  - This test case did not uncover any bugs.
- Submit the tracking log publishing form with a resolution percentage equal to that of a previous tracking log to check that errors are reported, as the resolution percentage must be monotonically increasing.
  - This test case did not uncover any bugs.
- Publish an `ACCEPTED` tracking log with correct attributes, checking that no errors or exceptions are reported and that it is displayed as published.
  - This test case did not uncover any bugs.
- Submit the tracking log publishing form with a resolution percentage lower than that of a previous one, checking that errors are reported.
  - This test case did not uncover any bugs.
- Submit the tracking log publishing form for an exceptional ("reclaimed") tracking log, with a status different from the one of the previously completed tracking log, checking that the appropriate error is reported.

  - This test case did not uncover any bugs.

- Publish an exceptional ("reclaimed") tracking log, checking that it is published without errors or exceptions and that it is displayed as published.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to publish a tracking log belonging to `agent1` as an unauthenticated user, as a user with a different realm, and as a different user with the same realm, checking that all result in an authorization failure.
  - This test case did not uncover any bugs.
- Request to publish a tracking log with ID -1, with ID `xxx`, with an empty ID, with no ID parameter, with the ID of an already published tracking log, and with the ID of a tracking log associated to an unpublished claim, to check that they all result in an authorization failure.
  - This test case did not uncover any bugs.
- Hack the `creationMoment` and `lastUpdateMoment` read-only fields of the publishing form, and set them to both empty values and values different from the ones initially shown, checking that no errors or exceptions occur and no changes are made.
  - This test case did not uncover any bugs.

2.2.6 Delete

Safe test cases recorded:

- Delete a tracking log, checking that it is correctly deleted without errors or exceptions and it no longer exists.
  - This test case did not uncover any bugs.

Hacking test cases recorded:

- Request to delete a tracking log belonging to `agent1` as an unauthenticated user, as a user with a different realm, and as a different user with the same realm, checking that all result in an authorization failure.
  - This test case did not uncover any bugs.
- Request to delete a tracking log with ID -1, with ID `xxx`, with an empty ID, with no ID parameter, and with the ID of an already published tracking log, to check that they all result in an authorization failure.
  - This test case did not uncover any bugs.

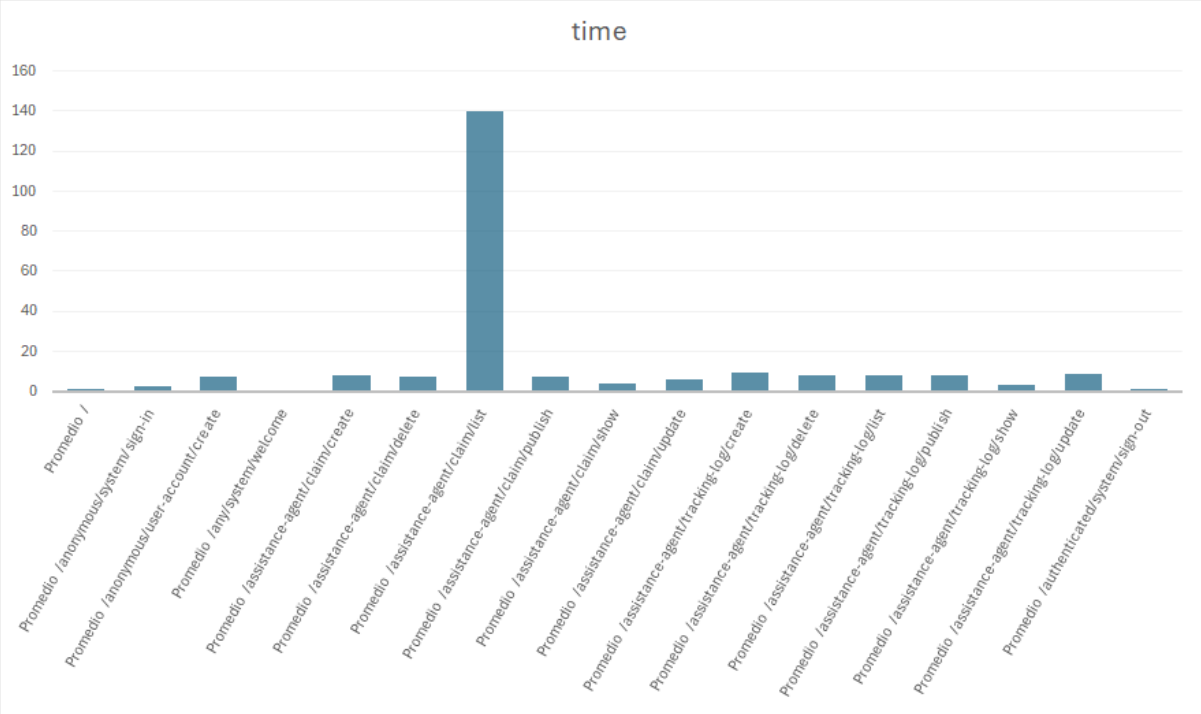

Figure 1: Average Lenovo Legion times

3. Performance Testing

In this chapter, the results of an analysis of the performance of the system under two different computers will be presented, including graphical representations of the time taken to serve each type of request in each computer, as well as a comparison between both machines to assess which one performed better.

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

- 2024 Lenovo Legion with an i7 CPU, 16GB of RAM and a 1TB SSD.
- 2024 HP Victus with an i7 CPU, 32GB of RAM and a 1TB SSD.

3.1. Lenovo Legion

After running the full test suite for both entities and analyzing the resulting system trace file, the average time to serve each request using the Lenovo Legion can be seen in the following bar chart:

As can clearly be seen in the chart, the Most Inefficient Request (MIR) is the one for the endpoint `/assistance-agent/claim/list`, taking an average of 140.3155615ms to serve in this computer.

Therefore, after performing a statistical analysis in Excel following the course indications:

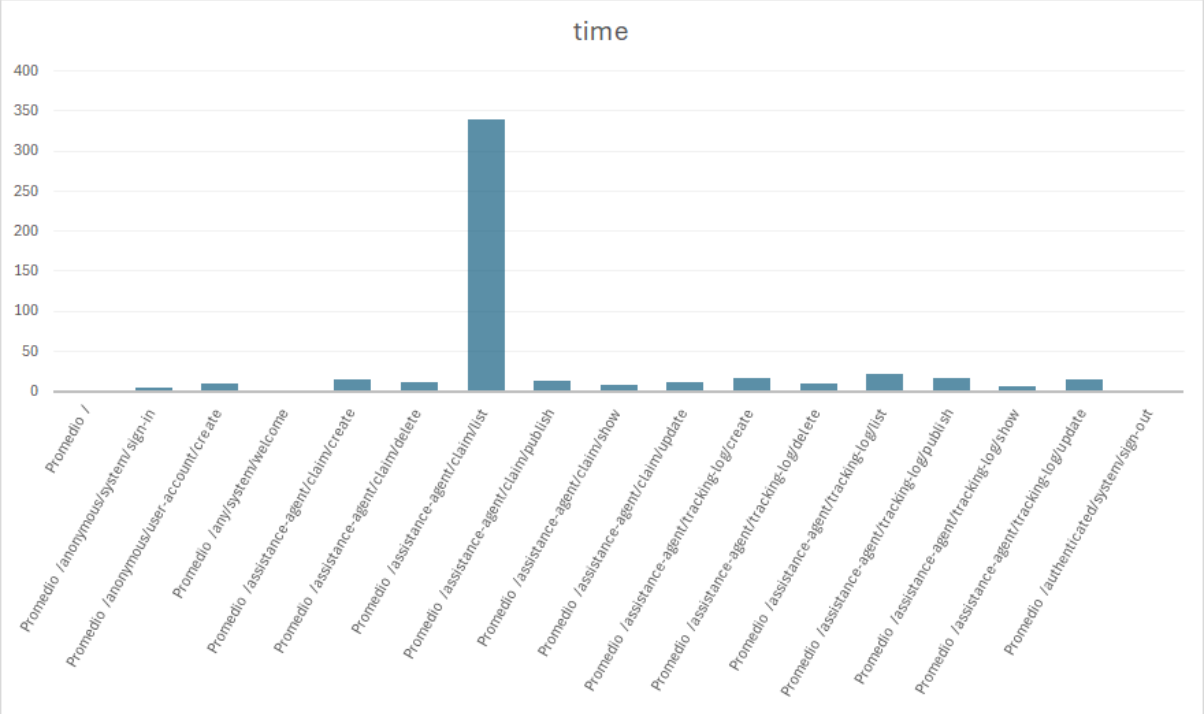

Figure 2: Average HP Victus times

- The average time for the Lenovo Legion is 18.62489497ms.
- The 95%-confidence level for the Lenovo Legion is 2.018938821.
- The 95%-confidence interval for the computer will therefore be:
  - [16.60595614, 20.64383379] in milliseconds.
  - [0.016605956, 0.020643834] in seconds.

3.2. HP Victus

The same process was performed for the HP Victus computer. The average times obtained were the following:

The MIR is the same as in the previous case (`/assistance-agent/claim/list`), however, it takes an average of 339.4687321ms to serve under this computer.

For this machine:

- The average time is 43.53889467ms.
- The 95%-confidence level is 5.138138918.
- The 95%-confidence interval is:
  - [38.40075575, 48.67703359] in milliseconds.
  - [0.038400756, 0.048677034] in seconds.

3.3. Comparison

In order to compare both machines, a 95%-confidence hypothesis contrast was performed. It can be supposed that, given the difference between the average times of both computers, the Lenovo Legion will be the most powerful of both. However, this must be numerically verified in order to provide a clear statement.

For that, a Z-test was first done in Excel following the course indications, in order to check if both average times could be compared. The Z-testing yielded a P-value of 0, meaning both averages could be compared in order to establish which machine was more powerful.

Therefore, as the average time for the Lenovo Legion was significantly smaller than that of the HP Victus, we can verify that it delivered the best performance out of the two computers when tested.

4. Conclusions

The functional testing process was instrumental in validating the design choices made for the system, as it confirmed empirically that the code written for the Acme ANS project is sufficiently secure to resist hacking attacks that may hamper the user experience. It also aided in the detection of bugs, which arose when recording test cases and were subsequently solved, and provided a source of truth regarding the intended behavior of the system, as every change performed should produce the same results as in the tests in order to ensure compliance with the requirements.

Furthermore, the performance testing process also holds a significant degree of importance in ensuring that the system delivers the best performance possible even when running under different machines. The performance analysis process also uncovered what the most inefficient request in the test suite was, in this case being the one for the claim listing (as it is the one with the most items being fetched from the database), as well as the most powerful computer of the two that were tested, which was the Lenovo Legion laptop, it being also the one that produced the lower average times for each request in general.

References

Intentionally blank