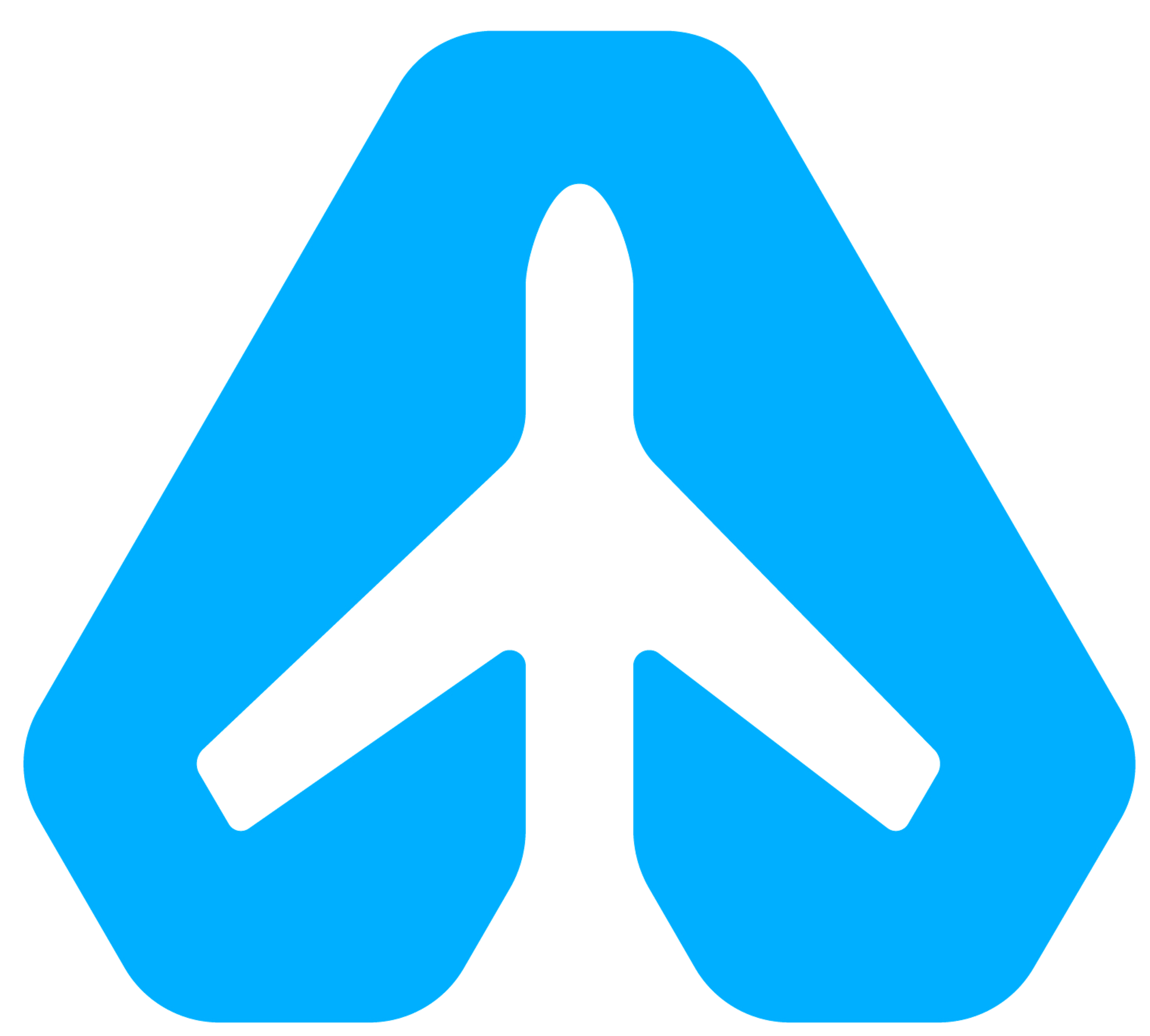
|  |
| --- |
| Acme AirNav Solutions, Inc. |
| **Testing Report** |
| http://github.com/Emili-115/DP2-Acme-ANS |

|  |
| --- |
| José Coronil Álvarez (joscoralv@alum.us.es)  Miguel García Vizcaíno (miggarviz@alum.us.es)  Javier Ignacio Milá de la Roca Dos Santos (javmildos@alum.us.es)  Emilio Manuel Vázquez Cruz (emivazcru@alum.us.es)  Group C1.028  5-26-2025 |



Contents

[Introduction 3](#_Toc199178969)

[Test Developed 3](#_Toc199178970)

[Claim Test 3](#_Toc199178971)

[Tracking Log Test 4](#_Toc199178972)

[Test Coverage 6](#_Toc199178973)

[Claim 6](#_Toc199178974)

[Tracking Log 6](#_Toc199178975)

[Performance Analisys 7](#_Toc199178976)

[Conclusions 10](#_Toc199178977)

# Introduction

This document is the testing report of the student 4’s deliverables in the Acme-ANS project, these are the features of the assistance agent’s management of claims and tracking logs. It includes a summary of the test cases developed for the features implemented, covering both expected use cases and security checks against unauthorized access. It also includes the test coverage achieved and a basic performance comparison between with and without indexes.

# Test Developed

The following is a list of the test cases implemented, grouped by feature:

## Claim Test

* **List-claim.safe:**This test the list of claims that are completed, this is that a tracking log with 100% percentage has been published, and the status of the claim is no longer PENDING. It also tests the list of claims that aren’t complete yet. There is no specific hack test for these features because it was checked in a general test that we will see later. No bugs were found
* **Show-claim.safe, Show-claim.hack:**These tests show the data of a claim. The safe one guarantees that the information shows correctly, in case that the claim is published or not. The hack one checks that, if the claim doesn’t exist or it doesn’t belong to the logged agent, the system rejects your request. No bugs were found
* **Create-claim.safe, Create-claim.hack:**These tests check the feature of creating claims. The safe one checks that the introduced data is valid or invalid in all the fields of the form. It also checks the validations of the claim, for example, some fields cannot have null value. The hack one guarantees that, in case of trying to introduce an invalid leg or an id that is not 0, the system rejects the request. There was found a bug in which, if you change the leg id to a non-existent leg, it shows you a panic instead of saying that null values are not acceptable. This bug was fixed successfully.
* **Delete-claim.safe, Delete-claim.hack:**These tests check the features of deleting claims. These tests ensure that, on the hand of the safe request, it deletes the claim successfully and in the hand of the illegal request, it rejects the request for non-existent claims or claims that are not property of the logged agent. It also checks if the id of the delete has changed in any means or if the requested claim to be deleted is in draftMode or not. In case of been published, the system rejects the petition to delete the claim.
* **Update-claim.safe, Update-claim.hack:**These tests check the features of editing existing claims. These tests are like the delete and publish test, because it guarantees that I can only edit existing claims that belong to the logged agent and that are not published yet. It also verifies that the id of the updated claim is still the same. It also, in the safe test, checks that the fields of the claim are edited to valid values, checking that the leg field is a valid leg, for example. The bug that was found in the create test, with the non-existent leg issues also appears in the publish and update feature. It was fixed successfully.
* **Publish-claim.safe, Publish-claim.hack:**These tests check the features of publishing claims, this is making a claim not editable and not deletable. These tests are almost the same as the update ones besides the feature and the fact that, when you publish a claim, all the fields of it become readOnly attributes.
* **Non-agent-claim.hack:**This test checks that only a user logged as an assistance agent can access all the features listed above. This is the reason why the list test doesn’t have a hacking test.

## Tracking Log Test

* **List-show-tracking-log.safe,List-show-tracking-log.hack:**These tests check the features of listing and show the tracking logs of a specific claim. The safe ones check that the data is correctly displayed, and the hacking ones check that the claim is valid (exists and belongs to the logged agent). Also, it checks, in the show features, if the tracking log is valid. The checking of the user logged as agent is checked in another test, just like the claim’s tests.
* **Create-tracking-log.safe, Create-tracking-log.hack:**These tests check the features of creating tracking logs. It guarantees that the claim in which we are creating the tracking log is valid, that the fields are correct and follow the validation of tracking log. It also prevents us from creating a tracking log with a different id than 0 and it also, because of the design of the application, prevents us from creating a tracking log if the previous one hasn’t been published, or if it has a 100% percentage progress.
* **Delete-tracking-log.safe,Delete-tracking-log.hack:**These tests check the feature of deleting a tracking log. It ensures that only non-published and valid tracking logs with the valid agent logged can procced with the request. Also, it checks if the id has changed. No bugs were detected.
* **Update-tracking-log.safe, Update-tracking-log.hack:**These tests check the feature of updating a tracking log. Like in the claim tests, the test checks that all the fields are valid and passes the validator of the tracking log. It also checks that the tracking log, agent and claim are valid. The id and the draftMode value are also checked.
* **Publish-tracking-log.safe, Publish-tracking-log.hack:**These tests check the feature of publishing a tracking log. The are some differences in the complexity of this test with the claim ones. In addition to all the tested areas in the update tests of tracking log, it also checks that the claim is published before publishing the tracking log. It also checks that if a tracking log with the 100% percentage is published, the status of the tracking log must edit the status of the claim.
* **Reclaim-tracking-log.safe, Reclaim-tracking-log.hack:**These tests check the reclaim feature. This feature means that, when a claim is completed, this is that has a published tracking log with 100%, a new process of tracking log can start if the client is not satisfied with the resolution of the claim. These tests checks that the field of the new tracking log are all valid, that the claim and agent are valid, that the id equals 0, and that the previous new tracking log is published and is not 100% completed. After a new tracking log reach 100%, the process end completely.
* **Non-agent-tracking-log.hack:**This test works exactly like the one in the claim test, checking that the user logged is an assistance agent that can access the features listed above.

# Test Coverage

The following is a summary of the coverage achieved for all relevant files

## Claim

* **Validator** (49,3%)  
    
  It appears that the validator is not very safe, because it doesn’t even reach 50% coverage, but that is because most of the validator validate things for the sample data, like the date being before the arrival of the leg. But this doesn’t validate in the test because the date is automatically set to be the actual date. This also happened with the status validation, but because the status change with the tracking logs, in which case it is correctly validated, it doesn’t go through that code.
* **Controller, CreateService, ListService, ShowService and UndergoingListService, UpdateService** (100%):  
  These cases were entirely covered by the tests.
* **DeleteService** (99,6%), **PublishService** (99,7%):  
  These cases aren’t completed because the code in which it checks if the id has changed, the possible explication could be that the condition (id==security), only gives false when id has change, but security uses the value in the URL, and because of that it never changes.  
  A screenshot of a computer

  AI-generated content may be incorrect.

## Tracking Log

* **Validator** (87,9%)  
  In this validator, the only thing that is not tested is the Date validation, and is because the Date in the tracking log features goes automatically, so it can not be changed.  
  
* **Controller, CreateService, ListService, ReclaimService, ShowService** (100%):  
  These cases were entirely covered by the tests.
* **DeleteService, UpdateService** (99,5%), **PublishService** (99,6%):  
  Is the same problem that in the claim test, the securityId never changes.  
  A screenshot of a computer

  AI-generated content may be incorrect.

# Performance Analisys

The performance of the project was analyzed according to the specifications both before and after indices were created for the relevant tables.

The times were recorded executing only the tests regarding assistance agent features. The exact same tests were run with both; the only difference is the indexing of the tables. The specific results can be seen on the Excel files in this folder.

**Before Indexes**

Before the indexes, the test resulted in a 95% confidence interval from 26,68ms to 29,255ms and a mean of 27,97ms. The costliest operation was Updating a tracking log with an average 46,08ms

A screenshot of a data sheet

AI-generated content may be incorrect.

**After Indexes**Before the indexes, the test resulted in a 95% confidence interval from 26,18ms to 28,855ms and a mean of 27,52ms. The costliest operation was Updating a tracking log with 46,215

A screenshot of a spreadsheet

AI-generated content may be incorrect.

**Comparison**The two sample z-test for means resulted in a P(Z<=z) two-tail value of 0.996, which is above significantly the alpha value for 95% confidence level (0.05). Thus, we cannot compare them because they are so equal that any difference is redundant. In this case, the right solution will be to check the process or the indexes to see if we can improve the process of the test to obtain a comparable difference.A table with numbers and a black text

AI-generated content may be incorrect.

# Conclusions

All required tests were written and executed successfully. The features behaved correctly in both valid and invalid scenarios. The tests helped identify bugs, which were subsequently fixed. Test coverage was achieved for all reachable lines in validators and services, with some exceptions explained in that section . Performance tests didn’t show a significant difference after indexing, likely due to the lack of optimization and the ineffective process of the test by the computer.