**Testing Report**

**Group**: C1.027  
**Repository**: <https://github.com/DP2-C1-027/AirNav-Logistics-C2>

### **Group Members:**

1. **Garcia de Tejada Delgado, Jose**
   * Role: Developer
   * Email: josgardel8@alum.us.es
2. **Peñaloza Friqui, Nora**
   * Role: Developer
   * Email: norpennfri@alum.us.es
3. **Niza Cobo, Manuel Jesús**
   * Role: Manager, Developer
   * Email: mannizcob@alum.us.es
4. **Gomez Claraco, Nicolas**
   * Role: Developer
   * Email: nicgomcla@alum.us.es
5. **Campos Diez, Lucia**
   * Role: Tester, Developer
   * Email: luccamdie@alum.us.es

04/07/2025

**Table of Contents**

1. **Executive Summary**
2. **Revision Table**
3. **Introduction**
4. **Contents**
   1. **Functional testing**
   2. **Performance testing**
5. **Conclusions**
6. **Bibliography**

**Executive Summary**

Testing plays a vital role in both designing and maintaining software projects. It helps development teams verify the quality and reliability of their system with greater ease and confidence. Additionally, testing ensures that new updates or modifications do not introduce unexpected flaws or security risks. In this report, student #4 will document the testing procedures carried out for requirements #8 and #9.

**Revision Table**

|  |  |  |
| --- | --- | --- |
| **Revision Number** | **Date** | **Description** |
| 1.0 | 27/05/2025 | Initial report not finished |
| 2.0 | 04/07/2025 | Final version indexes added |

**Introduction**

The first section covers functional testing, where individual requirements will be formally evaluated and organized by feature. The second section focuses on performance testing, employing statistical analysis and visualization tools to measure system responsiveness. Key metrics, such as request delay, will be assessed with a 95% confidence interval to determine statistical significance. Additionally, the findings will be compared against initial hypotheses to validate performance expectations.

**Contents**

* 1. **Functional testing:**

In the following paragraphs, I’ll summarize the implemented testcases, grouped by entity and feature, and what each file in the testsuite has been used on.

**Claim**

* **-List-Completed.safe :** We log in as agent1 and attempt to access our completed claims via the button in the menu. Then we do the same thing as agent3 (empty list**)**
* **List-Ongoing.safe:**We log in as agent1 and attempt to access our on going claims via the button in the menu. Then we do the same thing as agent3 (empty list**)**
* **Show.safe :** We log in as agent1 and access many of our claims, both published and unpublished.
* **Show-Leg.safe:** We log in as agent1 and access many of our claims one unpublished for example, once we are in the claim we try to access the leg details by the button. We repeat the process with published.
* **Create.safe** We log in as agent1 and attempt to create as many invalid claim as possible checking empty fields, outside of range, invalid format, and with valid fields, edge of range values and all the choices in the enumerates. We finish with a valid create.
* **Update.safe** We log in as agent1 and attempt to update as many invalid claim as possible checking empty fields, outside of range, invalid format, and with valid fields, edge of range values and all the choices in the enumerates. We finish with a valid update.
* **Delete.safe** : We log in as agent1 and attempt to delete some unpublished claims some with trackingLogs others empties.
* **Publish.safe :** We log in as agent1 and attempt to publish as many invalid claim as possible checking empty fields, outside of range, invalid format, and with valid fields, edge of range values and all the choices in the enumerates. We finish with a valid publish
* **List-Completed.hack** We try to access the endpoint like an unlogged user. We do the same as administrator. Both are caught by the framework.
* **List-Ongoing.hack** We try to access the endpoint like an unlogged user. We do the same as administrator. Both are caught by the framework.
* **Show.hack** We log in as agent2, and access one of our claims. We then change the id in the url to that of a non-existing, that of a claim that belongs to another agent, and invalid formats (“asdf”,”-123”,””, deleting the entire field). We also do this as an administrator and as an unlogged user
* **Show-Leg.hack** We log in as agent2, and access one of our claims. We then access to the related leg after that change the id in the url to that of a non-existing, that of a claim that belongs to another agent, and invalid formats (“asdf”,”-123”,””, deleting the entire field). We also do this as an administrator and as an unlogged user
* **Create.hack** We log in as agent1 and access the create form. We then use the inspect tool to assign a non-zero id to the hidden field (overwriting an existing claim), with an id with no claim associated, and invalid formats (“asdf”,”-5”,”as12345”,””). We also erase the entire field of the form. All attempts return non authorized exceptions.. We also attempt to access this form as an administrator and as an unlogged user.
* **Update.hack** We log in as agent1 and access one of our unpublished claim. We then use the inspection tool to set the hidden id field to an id with no claim associated, invalid formats, the id of a published claim we own, the id of an unpublished claim from another user. We also tamper with the same fields as in create.hack.
* **Delete.hack** We log in as agent1 and access one of our unpublished claim. We then use the inspection tool to set the hidden id field to an id with no claim associated, invalid formats, the id of a published claim we own, the id of an unpublished claim from another user. We also tamper with the same fields as in create.hack.
* **Publish.hack** We log in as agent1 and access one of our unpublished claim. We then use the inspection tool to set the hidden id field to an id with no claim associated, invalid formats, the id of a published claim we own, the id of an unpublished claim from another user. We also tamper with the same fields as in create.hack.

**TrackingLog**

* **List.safe :** We log in as agent1 and attempt to access our trackingLogs filter by claim via the button in the show claim view. Then we do the same thing but with a claim with no trackinLogs.
* **Show.safe :** We log in as agent1 and access many of our trackingLogs, both published and unpublished from diverse claims both published and unpublished too.
* **Show-claim.safe :** We log in as agent1 and access many of our trackingLogs one unpublished for example, once we are in the claim we try to access the claim associated details by the button. We repeat the process with a published one.
* **Create.safe :** We log in as agent1 and attempt to create as many invalid trackingLogs as possible checking empty fields, outside of range, invalid format, and with valid fields, edge of range values and all the choices in the enumerates. We finish with a valid create.
* **Update.safe** We log in as agent1 and attempt to update as many invalid trackingLogs as possible checking empty fields, outside of range, invalid format, and with valid fields, edge of range values and all the choices in the enumerates. We finish with a valid update.
* **Delete.safe** We log in as agent1 and attempt to delete some unpublished trackingLogs some in publish claim others in unpublished ones.
* **Publish.safe** We log in as agent1 and attempt to publish as many invalid trackingLogs as possible checking empty fields, outside of range, invalid format, and with valid fields, edge of range values and all the choices in the enumerates. We finish with a valid publish
* **List.hack** We try to access the endpoint like an unlogged user. We do the same as administrator. Both are caught by the framework.
* **Show.hack** We log in as agent2, and access one of our trackingLogs. We then change the id in the url to that of a non-existing, that of a claim that belongs to another agent, and invalid formats (“asdf”,”-123”,””, deleting the entire field). We also do this as an administrator and as an unlogged user
* **Show-claim.hack**We log in as agent2, and access one of our trackingLogs. We then access to the related claim after that change the id in the url to that of a non-existing, that of a claim that belongs to another agent, and invalid formats (“asdf”,”-123”,””, deleting the entire field). We also do this as an administrator and as an unlogged user
* **Create.hack** We log in as agent1 and access the create form. We then use the inspect tool to assign a non-zero id to the hidden field (overwriting an existing trackingLogs), with an id with no claim trackingLogs, and invalid formats (“asdf”,”-5”,”as12345”,””). After that we then use the inspect tool to assign a non-zero Claim id to the claim field and all the test describes before. We also erase the entire field of the form. All attempts return non authorized exceptions. We also attempt to access this form as an administrator and as an unlogged user.
* **Update.hack** We log in as agent1 and access one of our unpublished trackingLogs. We then use the inspection tool to set the hidden id field to an id with no claim associated, invalid formats, the id of a published trackingLogs we own, the id of an unpublished trackingLogs from another user. We also tamper with the same fields as in create.hack.
* **Delete.hack** We log in as agent1 and access one of our unpublished trackingLogs. We then use the inspection tool to set the hidden id field to an id with no claim associated, invalid formats, the id of a published trackingLogs we own, the id of an unpublished trackingLogs from another user. We also tamper with the same fields as in create.hack.
* **Publish.hack** We log in as agent1 and access one of our unpublished trackingLogs. We then use the inspection tool to set the hidden id field to an id with no claim associated, invalid formats, the id of a published trackingLogs we own, the id of an unpublished trackingLogs from another user. We also tamper with the same fields as in create.hack.
  1. **Performance testing:**

Testing before adding indexes

**A screenshot of a spreadsheet

AI-generated content may be incorrect.**

**Testing after adding indexes**

**A screenshot of a spreadsheet

AI-generated content may be incorrect.**

As we can see from the graphs, the slowest functions are the create and delete of the both entities. This stands to reason as those are the ones that undergo most validations relying on database queries specifically the create.

Looking at the graphs we can assume that the index improves the validations related with the delete features and affects the create only slightly.

Contrast hypothesis

A screenshot of a computer

AI-generated content may be incorrect.

The results of the z-test show a highly significant difference between the "Before" and "After" groups, with a large z-score of 3.62 and an extremely small two-tailed p-value of 0.000292 . Since this p-value is far below the standard significance level of 0.05 .

With a two-tail p-value of 0,00146 and a confidence level of .95, we are below the .05 p-value threshold for statistical significance, meaning we can make the claim that adding indexes did indeed make an improvement in our average response time compared to not having said indexes

**Conclusions**

Through rigorous testing, student #4 uncovered multiple authorization bugs and fixed a misimplemented security restriction. The report also includes performance comparisons—demonstrating that response times improved after refactoring the database with indexes.

**Bibliography**

Intentionally blank.