Analysis Report

Acme AirNav Solutions, Inc

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

Student #2:
María Quirós Quiroga, marquiqui@alum.us.es

May 26, 2025

Contents

Executive Summary

This document presents the analysis conducted for the requirements corresponding to Delivery 03 of Student #2. Only requirements that involved ambiguity, incomplete information or implementation challenges and required further decision and clarification were included. Each analysis record contains a verbatim copy of the original requirement, the identification of the issue, an evaluation of possible alternatives, the justification behind the final decision and references the validation performed by the lecturer.

The main issues addressed in this analysis include:

- Implementation of uniqueness constraints for attributes and the need for custom validators beyond basic persistence annotations.
- Clarification on how derived attributes like booking price and purchase moment should be computed and updated.
- Identifying natural identifiers for flights to ensure unambiguous booking references.
- Establishing the necessary conditions under which a booking can be published, including validation of associated passengers and credit card information.
- Proper handling and display of financial indicators in customer dashboards, especially regarding currencies and statistical calculations.

The conclusions and resolutions reached in this document are cemented in lecturer validations and are intended to guide a consistent and correct implementation of the requirements.

Revision History

Revision	Date	Description
1.0	2025-05-20	Initial draft
1.1	2025-05-26	Add requirement analysis

1. Introduction

The purpose of this document is to provide a comprehensive analysis of the requirements that required clarification for Delivery 03 of Student #2 [1]. This analysis aims to clarify ambiguities and address potential issues. Only requirements that required further evaluation have been included.

For each analyzed requirement, an analysis record has been documented, consisting of:

- A verbatim copy of the original requirement for reference.
- An description of the issues or ambiguities encountered.
- An analysis of potential solutions considered, including the advantages and disadvantages of each approach.
- A final decision made regarding the issue.
- A link to the validation performed by the lecturer, confirming correctness and applicability of the modifications.

This report is structured to ensure transparency in the decision-making process and serves as documentation for future reference.

2. Analysis Records

2.1. General

This section discusses some clarifications that are related the general implementation of requirements, not to any specific requirement.

2.1.1 First question

The requirement mentions that some attributes must be unique. We were not sure how to implement this uniqueness constraint. Several alternatives were considered:

- Alternative 1: Make use of the persistence annotation @Column.
  - Advantages: When using this annotation, the database may automatically check the uniqueness constraint. This is simpler and may be more efficient.
- Alternative 2: Implement a custom validator.

  - Advantages: Allows more control over the error messages.
  - Disadvantages: Will be more complex to implement and maintain.

Finally, the lecturers commented that the annotation only indicates to the database that it must create an index to speed-up the queries for uniqueness using these attributes. However, this does not validate that the attributes are actually unique, so a custom validator should be implemented as in Acme Jobs.

The validation performed by the lecturer can be found in the following links: 1, 2.

2.2. Requirement 4

The following requirement will be analyzed:

"A booking is a reservation made by a customer to purchase a flight, guaranteeing some seats on a specific itinerary and associating some passengers' details with the trip. The system must manage the following information for each booking: a locator code (unique, pattern "`^[A-Z0-9]{6,8}$`*"), a purchase moment (in the past), a travel class ("ECONOMY", BUSINESS"), and a price. Optionally, the system should record the last nibble of the credit card used for payment."*

2.2.1 First analysis

The requirement mentions that a booking has an associated price. The question is, should the price be a derived attribute?

- Alternative 1: Create a passenger number attribute that is multiplied by the cost of the associated flight to compute the price of the booking.
  - Advantages: The client can easily introduce the number of passengers and the system will automatically compute the price.
  - Disadvantages: We introduce an attribute that is not part of the requirement and may contain repeated information, as we already have the relationship between passengers and bookings stored in the `BookingRecord` entity.
- Alternative 2: Change the price attribute to be the price of the flight per person, which would be the flight cost.
  - Advantages: Easier to implement as there is no need to compute any operation with passengers.
  - Disadvantages: Does not reflect the total price of the booking.
- Alternative 3: Compute the price by multiplying the cost of the flight by the number of passengers associated to the given booking.

  - Advantages: It provides the total price of the booking without repeating any attributes.
  - Disadvantages: The price will be changing while the booking is completed and will only be final once it is published.

The third alternative seemed like the best option for us. Finally, the lecturers recommended to use the third alternative and compute the price by multiplying the cost of the associated flight and the number of passengers associated to the booking.

The validation performed by the lecturer can be found in the following link: 1.

2.2.2 Second analysis

The requirement mentions that a booking must store the purchase moment. This attribute should be computed automatically by the system but a doubt emerges, when should this attribute be computed?

- Alternative 1: Compute the purchase moment when the booking is created.
  - Advantages: The purchase moment is recorded as soon as the booking is created, and requires minimal tracking of the state of the booking.
  - Disadvantages: A booking may be created and never finalized or paid for, so the recorded "purchase moment" could be misleading.
- Alternative 2: Compute the purchase moment when the booking is published.
  - Advantages: Publishing a booking can be considered as confirming that the booking is finished as no changes can be made to it after it is published. So, the system would be storing the date in which the booking was confirmed.
  - Disadvantages: The purchase moment will not be computed until the booking is published and the user may have actually paid for it beforehand, as the purchase is not handled by our system.

After considering the alternatives, I think the best one would be to apply the second one, as it is more representative of the actual meaning of a "purchase date". The lecturer commented that any of the alternatives can be accepted. Finally, I decided to update the purchase date with each change to the booking entity, that is: setting it when creating the entity, then updating it with each update and finally setting it as definitive when the booking is published.

The validation performed by the lecturer can be found in the following link: 1.

2.3. Requirement 8

The following requirement will be analyzed:

"Operations by customers on bookings:

- *List their bookings.*
- *Show the details of their bookings and the associated passengers, if any.*
- *Create or update their bookings. Bookings can be updated as long as they have not been published. A booking can be published only when the last credit card nibble has been stored.*

"

2.3.1 First analysis

The requirement mentions that a customer should be able to show the details of their bookings and perform operations over them like creation and edition. However, that makes us wonder, how can we uniquely identify the flights that may be associated to the bookings to customers so that there is no ambiguity when selecting or referencing specific flights during booking operations?

- Alternative 1: Use the flight id.
  - Advantages: Simple and straightforward, does not require additional methods to display more data.
  - Disadvantages: Potentially misleading, as it may be difficult for users to associate a numeric id with a specific flight.
- Alternative 2: Use the flight tag.
  - Advantages: Simple and straightforward, does not require additional methods to display more data.
  - Disadvantages: The flight tag may be not be unique and does not provide a lot of information about the flight, which could lead to confusion and incorrect selections.
- Alternative 3: Use a custom string format *"%tag: %originCity → %destinationCity (Departure: %ScheduledDeparture, Arrival: %ScheduledArrival".*
  - Advantages: Provides clear and meaningful information about the flight, making it easier for users to identify each flight.
  - Disadvantages: Requires implementing an additional method to generate the string in the specified format.

- Alternative 4: Use a custom string format *"%originCity → %destinationCity (Departure: %ScheduledDeparture, Arrival: %ScheduledArrival".*
  - Advantages: Similar to Alternative 3, but excludes the flight tag, focusing only on essential details.
  - Disadvantages: Requires implementing an additional method to generate the string in the specified format.

After considering the alternatives, the students thought that the best solution was to implement alternative 3 or 4 depending on the level of detailed required by the customer.

The lecturers confirmed that flights do not have any natural identifiers that make them easy to differentiate in the context of the project. Therefore, they suggested using a combination of attributes from the legs of the flights. Similar to alternatives 3 and 4, the conclusion was to use a custom string format *"origin of first leg - destination of last leg"*. It was suggested that a derived attribute could be implemented on the flight entity to generate these strings, or that they could be generated at service level.

- Alternative 1: Generate a derived attribute to act as a natural identifier in the application for flights.
  - Advantages: The derived attribute can be used in all the services that require it by implementing it only once, minimizing duplication.
  - Disadvantages: If different features require different labels, one method may not be enough or may need parameters, which adds complexity.
- Alternative 2: Generate the labels at a service level in the unbind method of the services that require such label.
  - Advantages: Keeps presentation-specific logic out of the domain model. Can adapt to the context and change depending on the feature implemented easily.
  - Disadvantages: If multiple services need the label, logic might get duplicated.

Finally, it was decided to use the proposed custom string (*"origin of first leg - destination of last leg*) by implementing it as a derived attribute in the flight entity, as multiple services related to bookings and other student's requirements would make use of it.

The validation performed by the lecturer can be found in the following link: 1.

2.3.2 Second analysis

The requirement mentions that a customer should be able to publish a booking. This makes us question which are the conditions necessary to publish a booking?

- Alternative 1: A booking must have some passengers to be published.
  - Advantages: Ensures that every published booking contains essential information and forces the user to finish all critical steps before committing the booking.
  - Disadvantages: May prevent publishing in cases where partial information is acceptable.
- Alternative 2: A booking may be published without passengers or with draft mode passengers.
  - Advantages: Allows bookings to be finalized even when all passenger data isn't available yet.
  - Disadvantages: Published bookings may lack essential information, making them harder to process or analyze.

After considering the alternatives, the students thought that the best solution was to implement alternative 1, and only allow for booking publishing once the credit card nibble is stored and at least one passenger is associated to the booking.

The lecturers confirmed that bookings should have all their passengers published and should have at least one passenger for them to be correctly published, as this was an implicit restriction of the requirements.

The validation performed by the lecturer can be found in the following link: 1 and 2.

2.4. Requirement 15

The following requirement will be analyzed:

"The system must handle customers dashboards with the following indicators:

- *The last five destinations.*
- *The money spent in bookings during the last year.*
- *Their number of bookings grouped by travel class.*
- *Count, average, minimum, maximum, and standard deviation of the cost of their bookings in the last five years.*
- *Count, average, minimum, maximum, and standard deviation of the number of passengers in their bookings.*

"

2.4.1 First analysis

The requirements state that that the dashboard must store the money spent in bookings during the last year and the count, average, minimum, maximum, and standard deviation of the cost of the customer's bookings in the last five years. Both of these deal with the price attribute of the bookings, which uses the datatype `Money`. However, it is not clearly indicated if the system should show these indicators for all currencies.

- Alternative 1: Using one unique currency for all indicators, assuming all money attributes are in that currency when computing.
  - Advantages: Easy to implement, no need to manage currency conversions. Indicators can be compared as they are assumed to be the same currency.
  - Disadvantages: May be misleading as money in one currency does not have the same value as in a different currency.
- Alternative 2: Convert different money currencies and use one unique currency.
  - Advantages: Ensures that all financial indicators are computed on a common, real-world basis.
  - Disadvantages: Complicated, requires the use of an external API. The users lose the information regarding the currency used to pay.
- Alternative 3: Show the indicators in a table with one value computed for each currency used.
  - Advantages: Maintains information regarding the currency used. No need to rely on exchange rates.
  - Disadvantages: More complex presentation, having multiple rows or columns per indicator which may be overwhelming. Different currencies cannot be aggregated.

Finally, the lecturers concluded that both alternative 2 and 3 are good alternatives, however, alternative 3 requires careful implementation of the API calls not to get banned due to too many requests to the money exchange service, this approach is more involved that necessary. So, alternative 3 was implemented.

The validation performed by the lecturer can be found in the following link: 1.

3. Conclusions

The analysis recorded in this document provides a detailed evaluation of the requirements for Delivery 02 of Student #2 [1], identifying ambiguities and problems, assessing potential solutions, and making informed decisions to improve the understanding of requirements and progress towards the goal.

Each analysis was recorded in a structured manner, ensuring that the final conclusion was approved by the lecturer. By documenting the original requirements, issues encountered, considered solutions and final decisions, this report serves as a transparent reference of the decision-making process. Additionally, the lecturer's approval serves to validate the final decision made, ensuring correctness and feasibility.

The analysis records provide a clear justification for the decisions made. Moving forward, these decisions will serve as a solid foundation for implementing the requirements as originally intended by the lecturers, ensuring accuracy and minimizing unnecessary errors and confusion.

References

[1] *02 - Requirements – Student #2*, Enseñanza Virtual, 2025. Available on Enseñanza Virtual in the course inside the Project Statement folder.