# Requirements – Student #2

Please, fill in the following form, make sure that you have ticked the requirements that you consider fulfilled, save this document, **and attach it in its original format (.docx)** to every deliverable. Regarding your ID, please keep only four random digits and mask the others using an asterisk. **Please, note that this document must be edited with the desktop version of Word since the web version does not properly support forms.** Attaching this document entails that you are the authors of the work delivered, you have not cheated in any way, and you have read and understood the information delivered regarding the subject, with a special emphasis on the methodological guidelines and how your work is going to be graded. Make sure that your project works well with the latest version of the development framework.

|  |
| --- |
| **Group:** C2.032 |
| **Repository:** https://github.com/Javiergutpas/Acme-ANS-C2 |
| **Student #2**  **ID Number:** 49\*3\*\*\*\*X  **UVUS:**  javgutpas  **Name:**  Gutiérrez Pastor, Javier  **Roles:**  Developer, Tester |
| **Date:** Sevilla Julio 1, 2025 |

# MANDATORY Deliverable D01: introduction

## Information requirements

Intentionally blank.

## Functional requirements

1. Modify the anonymous menu so that it shows an option that takes the browser to the home page of your favourite web site. The title must read as follows: “〈id-number〉: 〈surname〉, 〈name〉”, where “〈id-number〉” denotes your DNI, NIE, or passport number, “〈surname〉” denotes your surname/s, and “〈name〉” denotes your name/s.

X

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Provide a link to your planning dashboard in GitHub to review the tasks, their current status, and your schedule.

X

# MANDATORY Deliverable D02: data models

## Information requirements

1. **Customers** are the people who purchase flights. The system must store the following data about them: an **identifier** (unique, pattern "^[A-Z]{2-3}\d{6}$", where the first two or three letters correspond to their initials), a **phone number** (pattern "^\+?\d{6,15}$"), a **physical address** (up to 255 characters), plus a **city** and a **country** (both up to 50 characters). Optionally, customers may have some **earned points** (up to 500k points).

X

1. 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.

X

1. A **passenger** is an individual who takes a flight and he or she must be registered in the corresponding booking. The system must store the following data about passengers: a **full name** (shorter than 256 characters), an **email**, a **passport number** (pattern “^[A-Z0-9]{6,9}$”), a **date of birth**, and, optionally, his or her **special needs** (shorter than 51 characters).

X

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

1. Produce assorted sample data to test your application informally. The data must include two **customer** accounts with credentials “**customer1**/**customer1**” and “**customer2**/**customer2**”. Create an additional customer account with credentials “**customer3/ customer3”** that represents a customer with only profile data.

X

## Managerial requirements

1. Provide a link to your planning dashboard in GitHub to review the tasks, their current status, and your schedule.

X

# MANDATORY Deliverable D03: implementing features

## Information requirements

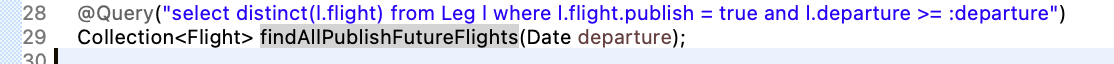
Intentionally blank.

## Functional requirements

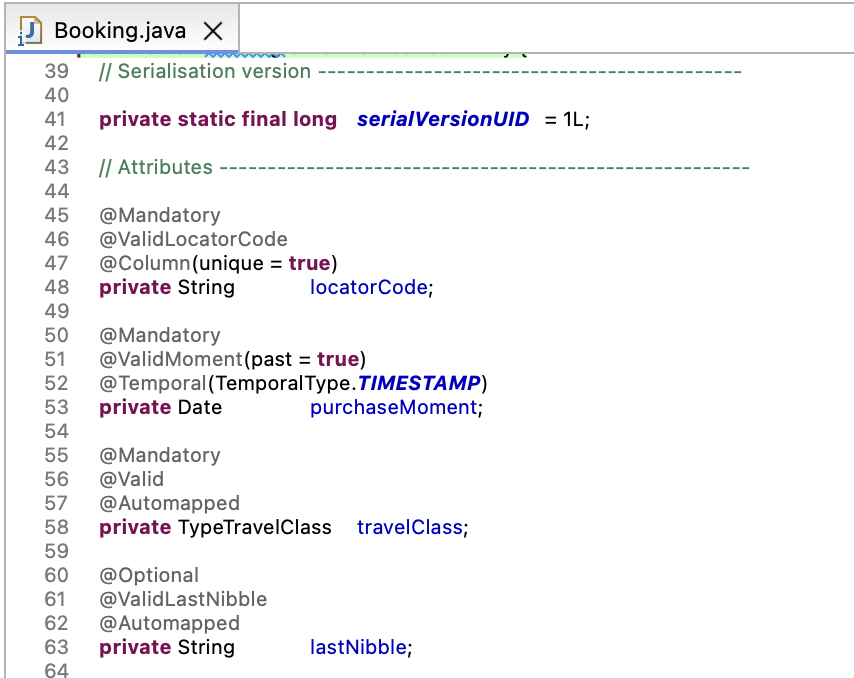
1. Operations by **customers** on **bookings**:

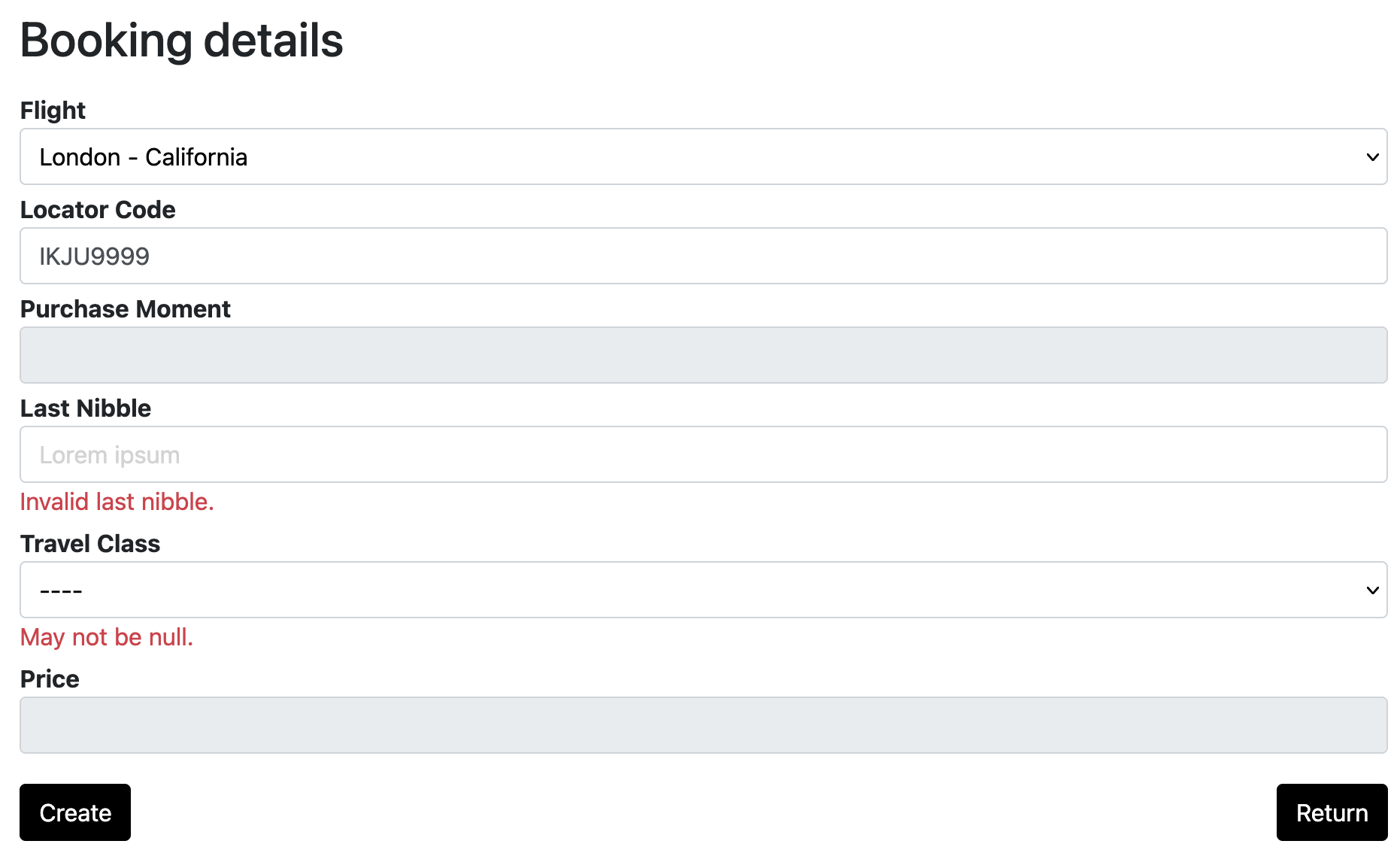
* List their bookings.
* Show the details of their bookings and the associated passengers, if any.
* Create or update their bookings. Bookings can be updated as long as they have not been published. A booking can be published only when the last credit card nibble has been stored.

Esta consulta para recuperar los bookings no es correcta, porque puede estar recuperando flights que ya hayan empezado si una de sus legs ya ha empezado.



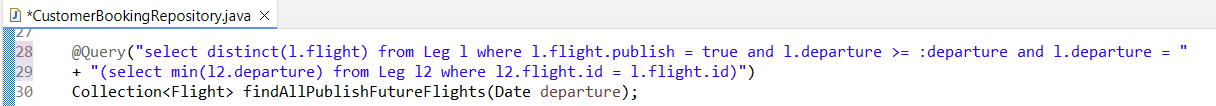
El last nibble es opcional, y está modelado como opcional, sin embargo, en el formulario, me pide introducir obligatoriamente ese dato:





Para solucionar el error relacionado con la consulta findAllPublishFutureFlights, he modificado su lógica de filtrado. La nueva consulta compara el atributo departure de la primera Leg de cada vuelo con la fecha de salida pasada como parámetro, garantizando así que solo se recuperen vuelos que comienzan en una fecha futura y que estén publicados.

La condición “… l.departure = (select min(l2.departure) from Leg l2 where l2.flight.id = l.flight.id)” asegura que se utilice únicamente la primera etapa (Leg) de cada vuelo para realizar esta comparación. A continuación, se muestra la consulta utilizada (en la captura adjunta se ha separado en varias líneas para facilitar su lectura, aunque en el código original se encuentra en una sola línea):



Con respecto al error relacionado con el atributo lastNibble de la entidad Booking, he realizado varias modificaciones para solucionarlo. En primer lugar, he eliminado la interfaz ValidLastNibble, que anteriormente se utilizaba para comprobar que el valor de lastNibble no fuese nulo. A continuación, modifiqué el atributo lastNibble, añadiéndole la anotación @ValidString(pattern = "^\\d{4}$"). De esta manera, se permite que este campo sea opcional en el formulario de creación de una reserva (Booking), validando únicamente que el formato introducido sea el correcto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

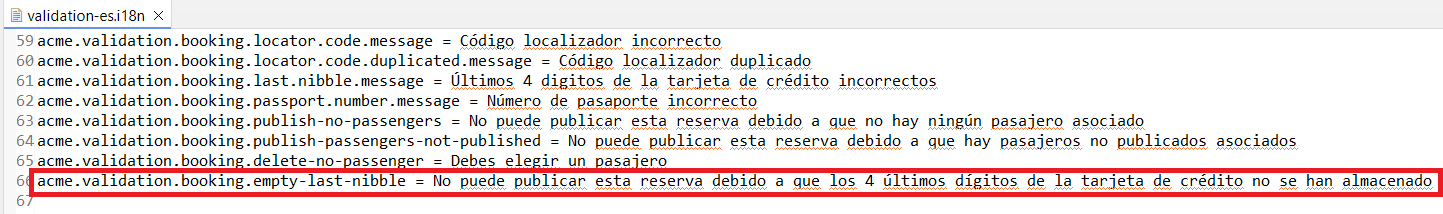
Seguidamente, he implementado de forma correcta el requisito que establece que no se puede publicar una reserva (Booking) si no tiene un valor almacenado en el atributo lastNibble (además de otras restricciones asociadas). Para ello, he modificado la clase CustomerBookingPublishService, concretamente el método validate, donde he añadido una nueva variable booleana que comprueba si el atributo lastNibble está vacío. Esta condición se evalúa junto con el resto de las validaciones necesarias antes de permitir la publicación de la reserva.

Texto

El contenido generado por IA puede ser incorrecto.

Por último, he creado los mensajes de validación correspondientes (tanto en inglés como en español) para el caso en el que se intente publicar una reserva (Booking) sin tener un valor almacenado en el campo lastNibble.

Captura de pantalla con letras y números

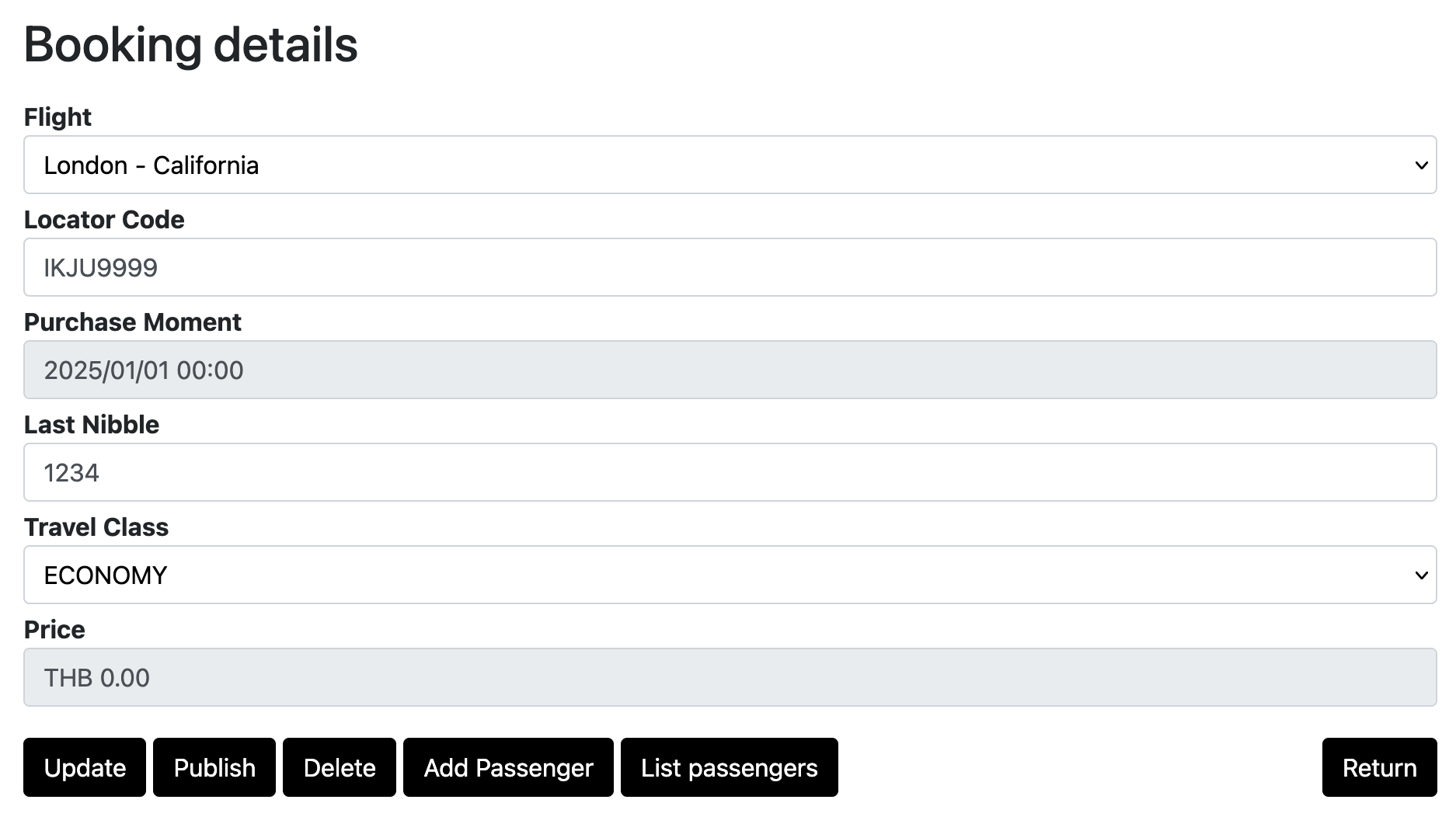
El contenido generado por IA puede ser incorrecto.

1. Operations by **customers** on **passengers**:

* List the passengers in their bookings.
* Show the details of their passengers.
* Create a passenger and record the information related to that passenger.
* Update a passenger as long as it has not been published.

No hay opción de añadir pasajeros o al menos yo no la encuentro en ninguna de las vistas:

Desde la vista de detalles de un booking, puedo ir a añadir pasajero o a listar los pasajeros:



Si hago clic en añadir pasajeros, ésta es la vista que obtengo:

A grey rectangular object with a white background

Description automatically generated

Como no hay ningun pasajero creado por este customer, pues no puedo añadir nada a través del desplegable. Si hago clic en listar a los pasajeros, va a una vista de listado que está vacía, porque aún no he dado de alta ningún pasajero, pero tampoco está el botón de create para crear un pasajero:

A screenshot of a computer

Description automatically generated

Aquí para la evaluación, el resto de requisitos no han sido evaluados.

Para solucionar este error, he realizado varios cambios en el código. En primer lugar, he modificado la clase CustomerPassengerListService. Específicamente, he eliminado la línea que añadía la propiedad global showCreate en el método unbind y la he trasladado al método load. De este modo nos aseguramos de que, en caso de que el Customer no tenga ningún Passenger creado, siempre se muestre el botón "Create", que permite acceder al formulario de creación de un nuevo pasajero. Esto se debe a que la propiedad showCreate se cargará de forma constante a través del método load.



En segundo lugar, he modificado la clase CustomerBookingRecordListService, eliminando la línea que añadía la propiedad global showCreate. Con este cambio, en la lista de pasajeros asociados a una reserva (Booking) ya no se mostrará el botón "Create" que permitía acceder al formulario de creación de un nuevo pasajero. Esto se debe a que, gracias a los cambios mencionados anteriormente, dicho acceso está garantizado mediante el botón "Create" situado en el listado general de los pasajeros (Passengers) creados por el Customer.



Finalmente, he realizado un pequeño ajuste en el archivo list.jsp correspondiente al listado de pasajeros, he añadido la condición de que aparezca el botón “Create” siempre y cuando la propiedad global showCreate sea igual a true.



## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Provide a link to your planning dashboard in GitHub to review the tasks, their current status, and your schedule.

X

# MANDATORY Deliverable D04: formal testing

## Information requirements

1. Create appropriate indices for your entities, if required.

X

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

1. Produce a test suite for Requirements #8 and #9.

X

## Managerial requirements

1. Provide a link to your planning dashboard in GitHub to review the tasks, their current status, and your schedule.

X

1. Produce a testing report.

X

# SUPPLEMENTARY I Deliverable D01: introduction

## Information requirements

Intentionally blank.

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

Intentionally blank.

# SUPPLEMENTARY I Deliverable D02: data models

## Information requirements

1. 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.

X

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Produce a UML domain model regarding the information requirements.

X

# SUPPLEMENTARY I Deliverable D03: implementing features

## Information requirements

Intentionally blank.

## Functional requirements

1. Operations by **anonymous principals** on user **accounts**:

* Sign up to the system and become a customer.

1. Operations by **customers** on user **accounts**:

* Update their profiles.

1. Operations by **administrators** on **bookings**:

* List the bookings in the system that are published.
* Show the details of the bookings (including the passengers).

1. Operations by **customer** on **dashboards**:

* Show their customer dashboards.

X

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Provide a link to a video in which you informally test requirement #8 and #9. Videos should not exceed 10 minutes in length and must be stored at the USE's facilities.

# SUPPLEMENTARY I Deliverable D04: formal testing

## Information requirements

Intentionally blank.

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Perform five mutations in your code and report on the results.

## Managerial requirements

1. Produce a lint report.

# SUPPLEMENTARY II Deliverable D01: introduction

## Information requirements

Intentionally blank.

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Produce an analysis report.

X

1. Produce a planning and progress report.

X

# SUPPLEMENTARY II Deliverable D02: data models

## Information requirements

1. The system must include a board to recommend something in the city and/or country of a given airport. **Recommendations** can be about experiences, activities, restaurants, accommodation or any other thing that a person may find interesting at the destination. A web service must be used to populate this entity with information about recommendations. Thus, the exact data to store depends on the chosen service, and it is the students' responsibility to define them accordingly. It is also the students’ responsibility to find the appropriate service; no implicit or explicit liabilities shall be covered by the University of Seville or their individual affiliates if the students contract pay-per-use services!  The students are strongly advised to ensure that the service they choose is free of charge.

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Produce an analysis report.

X

1. Produce a planning and progress report.

# SUPPLEMENTARY II Deliverable D03: implementing features

## Information requirements

Intentionally blank.

## Functional requirements

1. Operations by **customers** on **recommendations**:

* List recommendations related to final destinations in their bookings.

1. Operations by **administrators** on **recommendations**:

* Populate the database with recommendations somehow related to locations (e.g., related to cities or countries).

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Produce an analysis report.

X

1. Produce a planning and progress report.

# SUPPLEMENTARY II Deliverable D04: formal testing

## Information requirements

Intentionally blank.

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

1. Produce as a complete test suite as possible for Requirement #29 ensuring that the API is properly mocked.

## Managerial requirements

1. Produce an analysis report.

1. Produce a planning and progress report.