ENGINEERING DESIGN METHOD

# PHASE 1: PROBLEM IDENTIFICATION.

## Problem context:

An airline has problems with the order in the process of passengers entering and exiting their aircraft. Currently, this generates delays and stress in the crew and passengers, making it difficult to make decisions and assign seats.

## Definition of the problem:

The airline requires a system to improve order in the process of passenger check-in and check-out of its aircraft.

## Identification of needs and symptoms:

* The airline needs to improve the orderly process of passengers boarding and exiting its aircraft.
* The airline does not have a system dedicated to order the aforementioned process.
* The solution must take into account the order in which passengers arrive at the boarding lounge to enter the aircraft.
* The solution must take into account that first class passengers have special benefits to enter the aircraft, based on data such as accumulated miles, special attention required, third age, among others.
* The solution to the problem must take into account that the departure of the aircraft must be carried out in an established order based on the configuration of the company's aircraft.
* The solution to the problem must be efficient so that the system can support the handling of a significantly large amount of data with the minimum consumption of resources.

## Problem specification:

***Client:*** Airline.

***Users:*** Boarding agents, crew members.

***Functional requirements:***

* The system must allow:

RF1. Load passenger information.

RF2. Search for a passenger information.

RF3. Record the arrival of a passenger at the boarding lounge.

RF4. Show the order of entry of passengers.

RF5. Show the order of departure of the passengers.

***Non-functional requirements:***

* The system must:

*RNF1.* Be efficient in the search for passenger information.

*Specification of requirements:*

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (RF1)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | **RF1. Load passenger information.** | | |
| **Summary** | The system must allow loading passenger information for an airline flight. This information is stored in a database (for the first version, this database will be simulated with a plain text file). | | |
| **Inputs** | **Input name** | **datatype** | **Selection or**  **repetition condition** |
| none | none | none |
| **General activities necessary to obtain the results** | 1. The system is started. 2. The file is stored as the system database. | | |
| **Result or post-condition** | Passenger information for that flight is loaded to the program. | | |
| **Outputs** | **Output name** | **datatype** | **Selection or repetition condition** |
| none | none | none |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (RF2)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | **RF2. Search for a passenger information.** | | |
| **Summary** | The system must allow a boarding agent to look up a passenger's complete information with his or her identification number. | | |
| **Inputs** | **Input name** | **datatype** | **Selection or repetition condition** |
| passengerID | String | none |
| **General activities necessary to obtain the results** | 1. The boarding agent must enter the passenger's id. 2. It must be validated that the entered id is registered, otherwise, the user must be informed. | | |
| **Result or post-condition** | A message is displayed with the passenger's information or with an indication that the passenger is not registered in the system. | | |
| **Outputs** | **Output name** | **datatype** | **Selection or repetition condition** |
| passengerInfo | String | none |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (RF3)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | **RF3. Record the arrival of a passenger at the boarding lounge.** | | |
| **Summary** | The system must allow registering the arrival of a passenger at their designated boarding gate. | | |
| **Inputs** | **Input name** | **datatype** | **Selection or repetition condition** |
| option | int | none |
| passengerId | String | none |
| **General activities necessary to obtain the results** | 1. The boarding agent must select the option to record the arrival time of a passenger at the boarding lounge. 2. The boarding agent must enter the passenger's id to search his information (See RF2) 3. The boarding agent records the arrival time. | | |
| **Result or post-condition** | Passenger's arrival time at the boarding gate is recorded in the system. | | |
| **Outputs** | **Output name** | **datatype** | **Selection or repetition condition** |
| confirmationMsg | String | none |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (RF4)**

| **Name or identifier** | **RF4. Show the order of entry of passengers.** | | |
| --- | --- | --- | --- |
| **Summary** | The system must allow showing the cabin crew in which order passengers should board the plane, considering their order of arrival. Likewise, the system shall prioritize the boarding order of passengers by sections, starting with those farthest from the entrance gate to the one closest to it. | | |
| **Inputs** | **Input name** | **datatype** | **Selection or repetition condition** |
| option | int | none |
| **General activities necessary to obtain the results** | 1. The cabin crew in charge must select the option to show the order in which passengers will be allowed to enter the plane. 2. The order of entry to the aircraft must be organized, according to their priority. | | |
| **Result or post-condition** | Order in which passengers will be able to board the plane is shown. | | |
| **Outputs** | **Output name** | **datatype** | **Selection or repetition condition** |
| entryOrder | String | none |

**FUNCTIONAL REQUIREMENTS ANALYSIS TABLE (RF5)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | **RF5. Show the order of departure of passengers.** | | |
| **Summary** | The system must allow showing the crew member in charge in which order passengers should exit the plane, considering the plane's configuration. | | |
| **Inputs** | **Input name** | **datatype** | **Selection or repetition condition** |
| option | int | none |
| **General activities necessary to obtain the results** | 1. The boarding agent must select the option to show the order of departure of passengers. 2. The order of departure will be governed by the following conditions: those who leave first are those who are in the first rows, and for each row the order is established by proximity to the aisle or order of arrival as the last instance. | | |
| **Result or post-condition** | Order in which passengers will be able to exit the plane is shown. | | |
| **Outputs** | **Output name** | **datatype** | **Selection or repetition condition** |
| departureOrder | String | none |

# PHASE 2: COLLECTION OF THE NECESSARY INFORMATION.

To address this problem, it is necessary to use some programming concepts and tools. Below is some information that may be useful for the implementation of the system:

1. Plain text files: One option to emulate the database in early versions of the program is to use a plain text document that includes the travelers' data following a given format.
2. Data structures: It is necessary to choose appropriate data structures to store passenger information and to be able to search it efficiently. Some options are linked lists, binary search trees, hash tables, queues and priority queues.
   1. **Linked List:** A linked list is a dynamic data structure used to store a collection of items. This is a sequence of links containing elements where each link contains a connection to another link. Taken from: [Data Structure and Algorithms - Linked List (tutorialspoint.com)](https://www.tutorialspoint.com/data_structures_algorithms/linked_list_algorithms.htm)
   2. **Binary Search Tree:** A binary search tree (ABB) is a binary tree with the property that all elements stored in the left subtree of any node x are less than the element stored in x, and all elements stored in the right subtree of x are greater than the element stored in x. Taken from: [BINARY SEARCH TREES (ugr.es)](https://ccia.ugr.es/~jfv/ed1/tedi/cdrom/docs/arb_BB.htm)
   3. **Hash Table:** The hash table is a data structure that stores data associatively and uses an array as a storage medium. Each data value has its own unique index value, which makes data access very fast if we know the index of the desired data. Due to its fast insertion and lookup operations, the hash table is an efficient data structure for handling large amounts of data. The hash technique is used to generate a unique index where an item will be inserted or located in the hash table. Taken from: [Data Structure and Algorithms - Hash Table (tutorialspoint.com)](https://www.tutorialspoint.com/data_structures_algorithms/hash_data_structure.htm)
   4. **Queues:** A queue is a type of data structure that stores elements of the same type. The components of a queue are stored in a FIFO (first in, first out) behavior. Taken from: [Java Queue - Javatpoint](https://www.javatpoint.com/java-queue)
   5. **Priority Queues:** A priority queue is an abstract data type that behaves similarly to the normal queue, except that each element has some priority, i.e., the element with the highest priority would be the first in a priority queue. The priority of the elements in a priority queue will determine the order in which the elements are removed from the priority queue. Taken from: [Priority Queue (Data Structures) - Javatpoint](https://www.javatpoint.com/ds-priority-queue)
3. Sorting Algorithms: To sort passenger information according to different criteria (order of arrival, accumulated miles, special attention required, etc.), different sorting algorithms can be used, such as sorting by bubble, sorting by selection, sorting by insertion, among others.

In addition, for the development of the system, it is necessary to take into account the following information:

Aircraft configuration: The layout of the airplane is crucial to determine the departure order of passengers, which implies having knowledge of how rows and seats are distributed in the aircraft. Normally, seats are arranged alphabetically and in rows numbered from 1 to 20 or more, depending on the size of the airplane. Seat numbering follows the sequence of letters of the alphabet, from A to F, and are arranged from left to right, so that window seats are A and F, while aisle seats are C and D, and middle seats are B and E. Taken from: [How to know if you have touched the window seat on your next plane trip (lasexta.com)](https://www.lasexta.com/viajestic/consejos-viajeros/como-saber-tocado-asiento-ventana-proximo-viaje-avion_20220525628deb903fa5760001df644e.html#:~:text=Por%20norma%20general%20los%20asientos,al%20lado%20de%20la%20ventanilla.) .

Sectional revenue: It is common for airlines to give priority to first class passengers because they have paid for a more luxurious and exclusive travel experience (Taken from What is it like to travel First Class? | Skyscanner Spanish). However, for now the airline that has contracted us wants to maintain the system of calling by sections of the plane, starting from the farthest from the gate to the one closest to it.

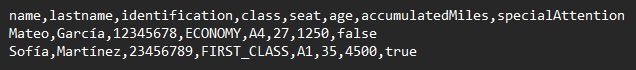
# PHASE 3: SEARCH FOR CREATIVE SOLUTIONS.

Different technologies and techniques can be used for the implementation of the airline's passenger check-in and check-out management system. However, it is important to take into account the efficiency and scalability of the system due to the large volume of data that will be handled in the future. Based on the above, we generated several ideas through a brainstorming session, a technique in which a group of people propose ideas freely and without restrictions, with the objective of generating creative and innovative solutions.

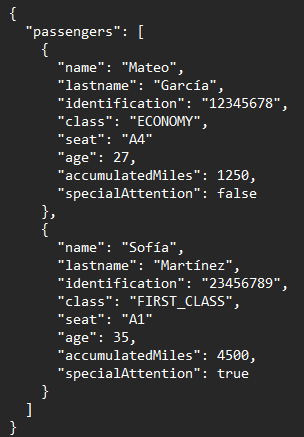
Database:

For the initial passenger load, a plain text file containing passenger information can be generated. The file format can be a CSV (comma-separated values) file, a JSON (JavaScript Object Notation) file, or a plain text file with fields separated by tabs or blanks.

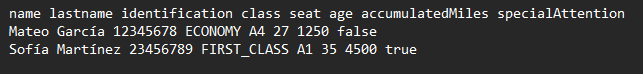
For example, the CSV format could have the following structure:

****

On the other hand, the JSON format could have the following format:



Finally, the text format separated by blank spaces could have the following structure:

****

This file can be read from the program and store passenger information in a data structure to reduce search time and optimize system performance.

Search for passenger information:

For the efficient search of passenger information, different data structures can be used:

1. Binary search trees: Passengers can be sorted based on their name or ID.
2. Hash Tables: Can be used with the passenger ID as the key and store all passenger information in one object.

Passenger check-in at the boarding lounge:

Different solutions can be used to check in passengers at the boarding lounge:

1. Manual check-in: Manual check-in is carried out by a designated airline staff member.
2. Fingerprint scanning: This technology would automate the passenger identification process, reducing the time required for manual check-in. It would also free up airline staff to handle other important tasks during the boarding process.
3. Drones equipped with cameras and image recognition software: This technology could provide an interesting alternative for identifying passengers in a crowd. Drones could be programmed to fly over the check-in area and capture high-resolution images that would be processed by image recognition software to automatically identify passengers.
4. Facial recognition technology: This technology allows passengers to be automatically identified by recognizing unique facial patterns. Passengers simply need to stand in front of a camera and their identity would be instantly verified by facial recognition software. This would reduce waiting time and improve the overall check-in experience for passengers.

Aircraft entry:

When boarding passengers on an aircraft, it is essential to take into account the configuration of the aircraft, which is divided into two sections: First Class and Economy Class. It is important to note that Economy Class passengers must enter the aircraft first, since their section is located at the rear of the aircraft (they are given priority only on a first-come, first-served basis). Afterwards, First Class passengers will be allowed access, whose priority is determined by several factors, such as assigned seat, need for special attention, advanced age, accumulated miles and order of arrival.

To carry out this process efficiently, a sorting algorithm, such as Merge Sort or Quicksort, can be used to prioritize the entry of First-Class passengers. These algorithms will sort passengers according to their priority and once sorted, they will be added to a queue to facilitate their entry. Another alternative is to use a priority queue that sorts passengers by priority and allows faster access to the next passenger.

Departure of the aircraft:

During the process of disembarking from an aircraft, it is important to prioritize the exit according to those passengers who are in the first rows and for each row the order is established by proximity to the aisle. In this order of ideas, first class passengers will be the first to leave the aircraft, since their section is located at the front of the plane.

To carry out the above process, the same alternatives can be implemented that were used for the entrance. Giving priority according to the seat where the passenger is located.

# PHASE 4: TRANSITION FROM IDEAS FORMULATION TO PRELIMINARY DESIGNS.

During phase 3 of the development process of the airline's passenger check-in and check-out management system, several ideas were generated through a brainstorming session. Next, in phase 4, preliminary designs based on the ideas generated in the previous phase will be carried out.

Database:

For the storage of passenger data, it was decided to use a plain text file with JSON format because this format has several advantages over other formats such as CSV or whitespace-separated text format.

First, the JSON format is easier to read and understand than plain text-based formats because it uses an object and attribute structure that is very intuitive for developers and computer systems. In addition, JSON objects can contain complex and structured information, which allows for greater flexibility in storing and processing data.

Another important advantage of JSON is that it is a highly interoperable format, which means that it can be used by a wide variety of systems and platforms without prior conversion.

Finally, the JSON format is highly serializable, which means that it is easily convertible into a machine-readable format and can be transmitted over the network efficiently.

Search for passenger information:

One of the ideas that is discarded is the use of a data structure based on binary search trees for searching passenger information. Although binary search trees are efficient data structures for searching and retrieving information with "O(log n)" complexity, they may not be the best choice for managing large volumes of data in real time, as their performance may be affected by search time and tree size. In addition, balanced trees must be maintained to ensure optimal operation, which can be costly in terms of resources.

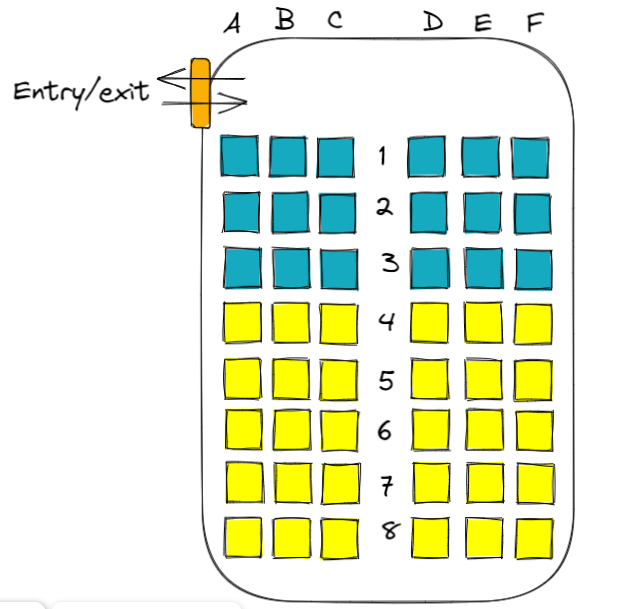
Instead, we opt for the use of a hash table that can use the seat number or passenger ID as a key and store all passenger information in one object. This is because hash tables are very efficient data structures for information retrieval, as they allow direct lookup and access to data in constant time, regardless of the size of the table, so their average complexity is "O(1)".

Passenger check-in at the boarding lounge:

Technologies such as facial recognition, fingerprint scanning and drones are very expensive and could create problems for the airline because of the security and privacy constraints they pose to users. Because of this, manual check-in looks like the most viable and simple option to execute, since it does not require a high investment in technology, is scalable and does not pose high privacy concerns for users.

Aircraft entry:

Before determining how to implement aircraft entry, an analysis of the seating layout of the company's aircraft was conducted. The aircraft consists of 8 rows, each with 6 seats divided by an aisle:



Based on this information, a decision was made as to how the entry of passengers into the aircraft would be carried out. Initially, the use of queuing algorithms was considered to speed up the check-in process. However, while this technique can be advantageous by prioritizing passenger entry quickly, it can be more complex and time-consuming to implement. Due to these disadvantages, this option was discarded.

Instead, a priority queue was chosen to organize the entry of passengers into the aircraft, as it provides fast and efficient access to passengers according to their priority, which can speed up the process of entering the aircraft. In addition, this alternative is simpler to implement.

Departure of the aircraft:

With respect to aircraft departure, the use of ordering and queuing algorithms has been discarded due to the fact that their implementation may be more time-consuming.

Instead, it has been chosen to use priority queuing to manage the departure of passengers from the aircraft, assigning them a priority according to their location in the aircraft. In this way, priority for departure is given to passengers closest to the aisle and in the first rows. This alternative is more efficient and agile in terms of time and resources, allowing a faster and more orderly exit of passengers from the aircraft.

# PHASE 5: EVALUATION AND SELECTION OF THE BEST SOLUTION

Depending on the solutions proposed to the problem, some criteria are presented that may be relevant to evaluate the ideas generated in the brainstorming phase for the implementation of the airline's passenger check-in and check-out management system:

**Criterion A: Efficiency -** Refers to the ability of the system to process and deliver the required data and services quickly and without interruption. Efficiency is measured in terms of speed of response or by the time complexity of operations on a scale of 1 to 5, with 1 being very low efficiency and 5 being very high efficiency.

**Criterion B: Scalability -** Refers to the ability of the system to grow and evolve without restrictions, and to meet future business or user needs without compromising its quality or stability. It can be evaluated on a scale of 1 to 5, with 1 being very low scalability and 5 being very high scalability.

**Criterion C: Development time -** Refers to the time to complete the development of a specific system or functionality. This criterion is affected by factors such as the complexity of the system and the urgency of the airline to have the system in place. It can be evaluated on a scale of 1 to 5, with 1 being too much time and 5 being too little time.

Evaluation of ideas, where the highest score is the best alternative:

Database:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Alternative** | **Efficiency** | **Scalability** | **Time of development** | **Total** |
| CSV | 4 | 4 | 4 | **12** |
| JSON | 5 | 4 | 5 | **14** |
| Text separated by whitespace | 3 | 3 | 4 | **10** |

Search for passenger information:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Alternative** | **Efficiency** | **Scalability** | **Time of development** | **Total** |
| Binary search tree | 4 | 4 | 4 | **12** |
| Hash table | 5 | 5 | 4 | **14** |

Passenger check-in at the boarding lounge:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Alternative** | **Efficiency** | **Scalability** | **Time of development** | **Total** |
| Manual registration | 3 | 4 | 5 | **12** |
| Fingerprint scanning | 5 | 3 | 3 | **11** |
| Drones equipped with cameras and image recognition software | 2 | 4 | 4 | **10** |
| Facial recognition technology | 4 | 3 | 3 | **10** |

Aircraft entry:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Alternative** | **Efficiency** | **Scalability** | **Time of development** | **Total** |
| Queues and sorting algorithms | 4 | 4 | 3 | **11** |
| Priority queues | 5 | 5 | 4 | **14** |

Departure of the plane:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Alternative** | **Efficiency** | **Scalability** | **Time of development** | **Total** |
| Queues and sorting algorithms | 4 | 4 | 3 | **11** |
| Priority queues | 5 | 5 | 4 | **14** |

Based on the above evaluation, the best alternatives are:

Database: JSON format.

Passenger information search: Hash table.

Registration of arrival of passengers to the boarding room: Manual registration.

Plane entry: Priority queue.

Departure of the plane: Priority queue.