|  |  |
| --- | --- |
| CLIENT | Airline |
| USER | crew member in charge |
| FUNCTIONAL REQUIEREMENTS | 1. Upload preliminary data 2. Register airport Check In 3. Show aircraft entry order. 4. Show aircraft exit order. |
| PROBLEM CONTEXT | A well-known airline selected us to solve its organization problem regarding the entry and exit to an aircraft, for that, the airline divides their passenger in two categories, normal and first class,  the entry of an aircraft attends a call by section paradigm where first in are the ones in the farthest section from the door, but it varies depending on the passengers of a specific fly, for normal passengers, when their section is called, they are rewarded by arriving early, so the earliest in that section enters first, in case of the first class, the section call still, but they one called first depend on the following characteristics: arrival time, miles earned, need for special attention, senior citizenship, number of suitcases, membership level and need for medical care, the more of this characteristics a passenger has it’s more likely to entry first in its section.  Lastly for the exit, they follow the paradigm but inverted, the first out are the ones closest to the door and to the hallway, if two passengers are the same distance from the hallway the one that arrived earlier goes first. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identificator | uploadData | | |
| Summary | The system must be able to load a set of data of the passengers | | |
| Input | Input name | Data type | Selection or repetition condition |
| usersData | File | File exists |
|  |  |  |
| General activities necessary to obtain the results | 1. Verify that the file exists 2. Read the file 3. Deserialize the read information 4. Upload the data in a efficient structure to save it | | |
| Result or post-condition | If file exists, all the file data is read and is uploaded to a efficient structure, else an exception is thrown | | |
| Output | Input name | Data type | Selection or repetition condition |
| N/A | N/A | N/A |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identificator | checkIn | | |
| Summary | The system must be able to retrieve the information of a passenger given its ID and register its arrival time | | |
| Input | Input name | Data type | Selection or repetition condition |
| userID | Long | Not null |
|  |  |  |
| General activities necessary to obtain the results | 1. Verify that the passenger exists. 2. Get user’s information 3. Get the arrival time 4. Register its chekIn | | |
| Result or post-condition | If the user is found, its information is retrieved and its chekIn is registered with its arrival time, else, a message is shown | | |
| Output | Input name | Data type | Selection or repetition condition |
| confirmMsg | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identificator | showEntry | | |
| Summary | The system must be able to organize he passengers in order to meet the call section paradigm and conditions for normal (arrival time) and first class (arrival time, miles earned, need for special attention, senior citizenship, number of suitcases, membership level and need for medical care) passengers, the priority for the first class are calculated arbitrarily, but more characteristics means more priority | | |
| Input | Input name | Data type | Selection or repetition condition |
| ListOfPassengers | List<Passenger> | Isn’t Empty |
|  |  |  |
| General activities necessary to obtain the results | 1. Evaluate the priority of each passenger 2. Get the passenger with highest priority 3. Register the passenger and delete it from the list 4. Repeat until the list is Empty | | |
| Result or post-condition | A list of passengers meeting call criteria defined | | |
| Output | Input name | Data type | Selection or repetition condition |
| orderedPassengers | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identificator | showExit | | |
| Summary | The passengers in their seats, the system must be able calculate its exit order, they are called from the closest to the door to the farthest, then from the closest to the hallway to the farthest, if two passengers are the same distance to the hallway, the first out is the one that arrived earlier | | |
| Input | Input name | Data type | Selection or repetition condition |
| ListOfPassengers | List<Passenger> | Isn’t Empty |
|  |  |  |
| General activities necessary to obtain the results | 1. Evaluate the priority of each passenger 2. Get the passenger with highest priority 3. Register the passenger and delete it from the list 4. Repeat from step 4 until the list is Empty | | |
| Result or post-condition | All the file data is read and is uploaded in a efficient structure | | |
| Output | Input name | Data type | Selection or repetition condition |
| N/A | N/A | N/A |