# Introduction

# Description

You will design a system for flight reservations of an airplane company. There are planes, cities, airports, flights, crew, customers, sales/reservations, staff, etc. You should consider flight history of customers to give promotions, connecting flights, cancelled flights and different classes within a flight (business, economy). Customers also should be able to cancel their flights but they must be penalized. Reservations done by customers have to be cancelled after a certain time. Moreover, customers should be provided to buy extra luggage with additional fee.

# Why and How a Databases is going to be used as a part of to that project?

-> Furkan

# Requirements

## Functional Requirements

### Customers

* Customers can view scheduled flights with information of price, promotion, sales, connection details and duration.
* Customers can view prices of desired flights and get promotions and sales
* Customers can make reservations for available flights and cancel existing ones.
* Customers can make payments of desired and available flights.
* Customers can return their purchased tickets with a penalty with respect to ticket type and remaining time to the flight.
* Customers can choose seat, class, meal for their purchased tickets.
* Customers can buy extra luggage for their purchased tickets.
* Customers can cancel the extra luggage they bought before 24 hours from the flight.
* Customers can view the current status of the flights that they have purchased a ticket for.

### Managers

* Managers can register new planes to system, and delete the existing ones.
* Managers can send planes for repairing and assign the returning ones as available.
* Managers can hire and fire staff such as pilots, hostesses, ground team etc.
* Managers can add airports.
* Managers can add routes between two existing airports and delete the existing ones.
* Managers can delete airports if there is no route assigned to them.
* Managers can create flights; change and cancel existing ones.
* Managers can set prices of current flights.
* Managers can apply promotions or sales to current flights or change/delete the current ones.
* Managers can create crews from existing and unassigned stuff.
* Managers can assign crews to scheduled flights if available or remove the current assignments.
* Managers can view current reservations and purchased tickets for flights.
* Managers can view the current status of the flights.
* **Cancel reservations Make reservations Sell extra luggage Cancel extra luggage Penalize customers (Bunlar bu adamın işi değil )**

### Salesperson

* Salespersons can view scheduled flights with information of price, promotion, sales, connection details and duration.
* Salespersons can sell tickets and choose class, seat, and meal for purchased tickets.
* Salespersons can cancel reservations.
* Salespersons can take back the purchased tickets and refund them with a penalty with respect to ticket type and remaining time to the flight.
* Salespersons can sell or cancel extra luggage.
* Salespersons can view current reservations and purchased tickets for flights.
* Salespersons can view the current status of the flights.
* **Cancel reservations, (gereksiz) Make reservations, (gereksiz)**

### Ticketing / Gate Agents

* Ticketing / Gate Agents can register the luggage.
* Ticketing / Gate Agents can cancel reservations.
* Ticketing / Gate Agents can assign seats for not checked in tickets.
* Ticketing / Gate Agents can view current reservations and purchased tickets for flights.
* Ticketing / Gate Agents can view the current status of the flights.

## Non-Functional

### Quick response time & Scalability

* The system is wanted to be fast and scalable as possible. Since the system may have big amount of data, the response time might grow bigger and bigger. To prevent a long waiting time and having a faster response, we will use multi-threading and vertical scaling for faster information retrieval and space.

**(Burda kimin ne yapamayacağını detaylandırayım mı bir sürü?)**

### Authentication

* Permissions of the actions of the users will be clearly specified and they will not be able to perform any extra action besides the ones that are given to them.
* Customers will not be able to sell tickets, register luggage, create airports or routes and hire personnel etc.
* Salesperson is a personnel in the system but it has its own boundaries as well. They will not have the same permissions which the Managers have. They can have common permissions with Gate Agents.
* Ticketing/Gate Agent personnel will mainly register customers to their flights and will be able to do the main work for paid tickets. For instance, they can register luggage, view the flight details for the purchased tickets and assign seats for customers. On a lighter note, they can have common permissions with Salesperson.
* Managers are the people who are in charge of the management system. They have all the permission from hiring new staff to arranging all the flights, planes and even airports in the system.
* All staff can also be Customers.

### User friendly

* The system will be easily accessible and friendly, by meaning the user interface.
* Every user will be limited to see according to their permissions. For instance, Customers or Salesperson will not be able to see how Managers can affect the system. By designing the system in this way, every user will be dealing with only the permissions they are permitted. They will not be confused by seeing what are irrelevant to them.

### Accurate Data Distribution

* The system will not overwrite any data under any circumstances unless needed. When needed, the data, which is wanted to be changed, will be removed and new data will be entered to the system.
* No data will be lost during the actions. Flights, airports, tickets and sales will be dependent on each other so there will be a warning if any of these are missing in an action. By this way, possible data losses caused by the users will be prevented.

## Pseudo

* **MySql\*\* will be used**
* **Web site can be created with HTML, JavaScript, CSS, HTML 5, Php technologies. (Angular JS, Bootstrap)**

## Limitations

* **The plane type will be determined according to the flight time.**
* **An airport will be added after a city is added.**
* **A route can be added after an airport is added.**
* **A route cannot be deleted if there is a flight**
* **An airport cannot be deleted if there is a route.**