Project Description

Airline Seat Reservation - ASR

Objective

The goal of this project is to implement a seat reservation system for an airplane consisting of:

- A relational database for data storage
- A web-based frontend
- A computing backend implemented in Python

Requirements

The system should offer the following functionality. We list the minimal functionality required for a passing grade marked with **[!]** and several options to improve the system and achieve a better grade for the course marked with **[*]**. You are welcome to include other useful features beyond our suggestions to further improve the system.

Input Data

- The system must be able to read data from a text file "chartIn.txt" that represents a seat chart in the following format (example) into the database:

```
    A   B   C   D   E   F
1   A   B   C   D   E   F
2   A   B   C   D   E   F
3   A   B   C   D   E   F
4   A   B   C   D   E   F
5   A   B   C   D   E   F
6   A   B   C   D   E   F
7   A   B   C   D   E   F
8   A   B   C   D   E   F
9   A   B   C   D   E   F
10  A   B   C   D   E   F
```

  The first row represents the headers for the columns that represent seats; the first column represents the seat rows in the airplane. Every plane has an even number of seats, equally distributed to two sides, so it is easier to identify window, middle and aisle seats. In the example above, the airplane has three seats on each side. Seats A and F represent window seats, B and E middle seats, C and D aisle seats. **[!]**
- The system supports different seat classes, such as seats in first, business, and economy class and emergency seats. **[*]**

- The system supports assigning and displaying prices for different seats. **[*]**
- The system supports arbitrary seat layouts, e.g., A B C D E F G H I. For each seat, the system can store and display whether it is a window, middle or aisle seat and where the aisles are. **[*]**
- Admin users can create seat layouts via a graphical web-based user interface instead of reading them in via a text file. **[*]**

Login, Logout

- Users must be able to login into the system via the web interface. **[!]**
- The database should contain at least 4 user accounts with the following information: userID, name, user_name, password. Only these users are allowed to enter the system. One of the users should have admin privileges in your system (explained later). **[!]**
- The system offers a logout button. When users click it, the system displays a thank you message and quits after a timeout period. **[!]**
- The system offers a separate input for quitting the program, aside from the logout button. **[*]**
- The system allows users to create and delete accounts via the web-interface by specifying a username and password. **[*]**
- The system allows users to create and delete accounts via the web-interface by specifying an email and password. For account creation and deletion, a confirmation mail is sent to the specified email address. The user needs to confirm receiving the mail, e.g., via a token or clicking a URL. **[*]**

Display Seats

- The system displays a basic seat chart consisting of only letters grouped into two equally-sized groups for left and right, pie symbols and whitespace between the groups represent the aisle, e.g., A B C | | D E F. **[!]**
- The system shows available seats by showing the seat letter and occupied seats using X, e.g., X X C | | X E F. **[!]**
- The system displays a graphical representation for seats and uses color coding or dedicated symbols to represent available and occupied seats. **[*]**
- The display supports advanced features, such as seat categories (first, business, economy class), emergency seats, and the cabin layout (wings, exits, kitchen, bathrooms), seat prices, etc. **[*]**

Reserve/Cancel Seats

- Users can reserve seats by entering the seat number, e.g., 4A into a dialog field. If the seat is already occupied, an error message is displayed. Every action should be confirmed by the user before being committed. Once confirmed, the information is automatically updated in the database and the interface. Only the admin user can cancel a reservation. After a reservation is canceled, the seat becomes available, again **[!]**.
- All users can reserve or cancel seats anytime via clicking in the graphical user interface **[*]**
- The system sends a conformation email for reserved seats **[*]**

Statistics

- The system offers a statistics area exclusively for admin users. The options in this area or menu should at least include the following statistics that can be saved into a text file **[!]**
    - Number and percentage of available seats
    - Number and percentage of reserved seats

    - List of seats that are available
    - List of seats that are not available
    - Number of users in the system with their information, except for their password
- The system offers the above-mentioned statistics as charts. **[*]**
- The system offers additional statistics for advanced features, such as seat categories, seat prices, cashflow from seat reservations etc. **[*]**

Help

- The system offers a help page that describes how to use the system and all its options **[!]**

Coding Practices (all mandatory)

- During the final presentation, enable the teaching team to test a working prototype, either on your machine or better running online.
- Structure your source code appropriately into classes according to OOP principles.
- Adhere to the PEP 8 Style Guide and best practice guidelines
- Handle errors properly so the system does not crash.
- Include unit tests for all methods that read in data from external sources or user input and all methods that manipulate data.