Functional Requirements Specification

# 1. Introduction

1.1 Purpose

This paper provides as an outline for the planning and building of a seat reservation application for Apache Airlines, in particular for their new fleet of Burak757 aircraft. The application is designed for airline employees to handle seat bookings, verify booking clarity, and provide customers with the best possible experience.

The main purpose of this system are:

1. To allow users to smoothly reserve, cancel, and confirm seat availability.
2. To properly show the aircraft's seating arrangement.
3. To prevent invalid or duplicate reservations.

This document outlines the functional requirements (the system's expected actions) and non-functional requirements (the system's performance goals). It includes assumptions on the system's use and constraints that influence its development.

# 2. Functional Requirements

2.1. Requirement 1: Verify Seat Availability

Description:

This function enables the user to input a seat number (e.g., "12B") and check its current status. The system are required to:

* Confirm that the seat number is found inside the aircraft design ( rows 1–80, columns A–F).
* Determine the status of the seat:
  + Free ("F") – can be reserved.
  + Reserved ("R") – previously booked.
  + Unavailable ("X" or "S") - aisle or storage space; not available for reservation.
* Provide an informative message to the consumer, such as:
  + “Seat 12B is available for reservation.”
  + “Seat 12B is an aisle seat and is unavailable for reservation.”

This enables employees to directly confirm availability before trying to reserve a seat.

2.2. Requirement 2: Book a seat

Description: To book a seat for a client, the user can input the number of the seat . The system needs to:

* Check the seat number.
* Verify that the seat is not reserved or classified as aisle or storage.
* Change the seat condition to "R" (reserved) from "F" (free).
* Display a message verifying the reservation:
  + "The reservation for seat 21B has been completed."
* If a seat has been reserved or unavailable, the system need to provide the proper error message which is:
  + "We have already reserved seat 21B. Select a different seat please.”

By doing this, errors and duplicate reservations are avoided during peak times, such as check-in time.

* 1. Requirement 3: Free a booked seat

Description:  This feature allows employees to cancel a seat booking if a client changes their plans or the reservation was incorrect. This system is going to:

* Request the user to input their seat number.
* Validate if the seat is already reserved ("R").
* When the cancellation is verified, change it back to "F" (free).
* Give verification as follows:
  + "Booking for seat 29E has been cancelled."
* If the seat is already available, the system will inform the user.
  + "Seat 29E is not booked."
* This keeps the seating arrangement up to date and correct.
  1. Requirement 4: Display Booking Details

Description: This function displays the current seating arrangement for the Burak757 aircraft. It is going to:

* Show all seat rows (1-80) and columns (A-F).
* Show the status of each seat:
  + "F" - free.
  + "R" - booked.
  + "X" - aisle (not bookable).
  + "S" - storage (not bookable).
* Use right layout for easier reading. Example:
  + Row 10: F F R X F F
  + Row 11: R R F X S S

This illustration helps make it simpler to arrange, particularly when handling group reservations or last-minute adjustments.

* 1. Requirement 5: Display Menu and Exit the application

Description: When the application begins operating, it must display a menu including five primary options:

1. Check seat availability.
2. Book a seat.
3. Free a seat.
4. Show booking status.
5. Exit program.

* The menu will be:
  + Continue returning after every action until the user chooses "Exit."
  + When an invalid input is entered, such as "9" or "hello," the following is displayed:
    - "Wrong decision. Enter a number between 1 and 5 please.”
* This menu structure makes the application more dynamic and user-friendly, even for those with basic computer knowledge.

# 3. Non-functional Requirements

3.1 Performance

The application has to effective and fast. Here are some specific goals:

1. Providing a response to each user activity (checking, booking, freeing, and displaying seats) in less than one second.
2. Able to manage a maximum of 480 seats, which is80 rows multiplied by 6 seats, without experiencing any lag.
3. avoiding shuts down, even when users are switching between tasks in a short amount of time.

This confirms that the performance is perfect, particularly during times of heavy request, such as when passengers are checking in or they are boarding their flights

3.2 Usability

 The program needs to be simple to use, even for airline employees who have basic understanding of information technology. It should be able to:

* The program is executed in a terminal or console window, and there is no special setup required.
* It is important to provide clear instructions such as:
  + “Please provide your seat number, such as 10A.”
* The result should be understandable, and the formatting should be organised.
* Display informative messages after each and every activity.
* Avoid technical slang or confusing errors.
* The system need to provide the user with guidance rather than crashing in case they type anything incorrectly.
  1. Security

The current version has to maintain the security of the system even if it does not need logins or internet connection by:

* avoiding reservations for seats given as "X" (aisle) or "S" (storage)
* Preventing any attempt to book a seat that has already been booked
* confirming that seat numbers fit to the "row number + column letter" pattern (for example, "10A").
* Ignoring inputs like "ABC," "911a" or "Null" that can cause the system to break down

These checks prevent errors or crashes caused by incorrect input and helps protect clean data.

* 1. Maintainability

It should not be hard to make changes to the application in the future. So it have to be clear for the user(well commented in python), Apply functions and an organised layout to simplify the implementation of elements such as references or database storage in Part B, and Quick ways of fixing any errors.

# 4. Assumptions and Constraints

4.1 Assumptions

* All Burak757 aircraft use a same seating arrangement including 80 rows with 6 seats per row, designated A–F.
* The application will just be used by Apache Airlines employees, not by consumers.
* Employees will use the system on desktop or laptop computers which come with a keyboard.
* The system will be used in offline mode, which means that there will be no internet or cloud connection.
* This version will not have any permanent data storage (bookings will not be saved once the program has closed).

4.2 Constraints

* The system only operates in a terminal or console environment.
* All of the reservations disappear when the application is stopped since the data is only kept in the random access memory (RAM).
* There is no GUI or graphical seat map, only a text-based interface.
* It is required to be created only using Python, without any additional libraries.
* Reservation reference numbers, client information, and database functionality will only be provided in Part B.