Functional Requirements Specification

# 1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for the Apache Airlines Seat Booking System. The system is being developed using an Agile approach, with iterative delivery of features. This system will allow users to view, book, and cancel seats on a flight, while storing relevant booking and passenger information.

# 2. Functional Requirements

2.1 Requirement 1: Seat Matrix Display

Description:

The system must display the seat layout of the aircraft, using specific character representations:

* "F" for Free
* "R" for Reserved
* "X" for Aisle
* "S" for Storage

This layout must be updated and shown each time a booking or cancellation is performed.

2.2 Requirement 2: Check Seat Availability

Description:

Users must be able to input a seat identifier and receive feedback on whether the seat is available (F) or already reserved (R). The system should validate input format and restrict access to invalid or non-seat entries.

2.3 Requirement 3: Book a Seat

Description:

Users should be able to book a seat by entering a valid seat ID. The system must confirm that the seat is currently free before reserving it. Upon booking, a unique 8-character alphanumeric booking reference must be generated and linked to the booking.

2.4 Requirement 4: Cancel a Booking

Description:

The system must allow users to free a previously booked seat. Once cancelled, the seat status should change back to "F" and all associated passenger details must be removed from the database.

2.5 Requirement 5: Store Passenger Information

Description:

When a booking is made, the system must store the passenger’s information including:

* First name
* Last name
* Passport number
* Seat number
* Booking reference

This data must be stored in a persistent SQLite database and be retrievable as needed.

2.6 Requirement 6: Search by Booking Reference

Description:

Users must be able to enter a booking reference and retrieve the associated passenger and seat information. If the reference does not exist, the system should return an appropriate error message.

# 3. Non-functional Requirements

3.1 Performance

* The system should return results for seat availability or bookings within 2 seconds.
* Booking reference generation and seat layout updates should occur in real time.

3.2 Usability

* The system will be menu-driven, allowing users to interact via simple numbered options.
* Inputs must be validated to prevent invalid actions.
* Error messages should be clear and instructive.

3.3 Security

* Booking references must be unique and non-predictable.
* Passenger data stored in the database should be securely handled.

# 4. Assumptions and Constraints

4.1 Assumptions

* The system is being developed and maintained by a single developer. All roles (Product Owner, Scrum Master, and Developer) are assumed by the same individual.
* The development process follows an Agile methodology, meaning that system requirements may evolve over time and will be refined through continuous feedback.
* Each sprint cycle is assumed to last approximately 2–3 days and will focus on one functional module at a time.
* It is assumed that user interface adjustments will be made iteratively based on tutor guidance or user testing feedback.
* The system is designed to run in a single-user environment. Multi-user access, concurrency handling, and session management are not within the current scope.
* Users are assumed to have basic knowledge of command-line operations and can interact with text-based menu systems.
* The aircraft seat layout is assumed to be fixed and static. Seat rows, labels, aisles (X), and storage areas (S) will not change after initialization.
* All seat data is assumed to be correctly initialized before use, with predefined values representing Free (F), Reserved (R), Aisle (X), and Storage (S) statuses.

4.2 Constraints

* The system must be developed in Python and run within a command-line interface environment.
* SQLite will be used as the only supported database for storing and retrieving passenger and booking data.
* Seat statuses must be represented using ASCII-style characters:

"F" for Free, "R" for Reserved, "X" for Aisle, and "S" for Storage.

* All booking references must be generated as unique 8-character alphanumeric strings.
* The entire project (including development, testing, and documentation) is constrained to a 17-day timeline.
* Bug fixes will not be deferred between sprints in order to maintain system stability and ensure smooth progress.
* Any new requirements discovered during the process must be added to the product backlog and assigned to a future sprint based on priority and time availability.