Technical Specifications Document

1. Title Page

- **Project Name**: Airline Booking System
- **Version**: 1.2
- **Date**: November 7, 2025
- **Author(s)**:
  - Rey Jesus M. Teves
  - Jan Chelsea Lavaro
  - Darwin Besorio
  - Rex Bugcalao
  - Cristino France Madali
  - Manuel P Buenviaje II

2. Table of Contents

3. Introduction

- Purpose
  To develop a basic airline booking system that enables users to search for flights, book tickets, purchase add-ons, and manage their bookings efficiently.
- Scope
  The MVP includes flight search, booking, ticket management, payment processing, and the purchase of key add-on services (meals, shop items, baggage fees).
  It excludes advanced features such as seat selection, loyalty programs, or multi-city bookings.
- **Definitions, Acronyms, and Abbreviations**
  - **PNR**: Passenger Name Record
  - **API**: Application Programming Interface

- **ERD**: Entity-Relationship Diagram
- **MVP**: Minimum Viable Product

4. Overall Description

- Product Perspective
  The system is a standalone web application aimed at customers and airline staff to manage flight bookings and associated purchases.
- **Product Functions**
  - Flight search by date, origin, and destination
  - Booking creation and confirmation
  - Ancillary service purchase (meals, shop items, baggage fees)
  - User registration and login
  - Payment gateway integration for ticket and add-on purchase
  - Booking viewing and cancellation
- **User Classes and Characteristics**
  - **End Users**: Customers searching and booking flights, and purchasing add-ons
  - **Admin Users**: Airline staff managing flight schedules, inventory, and bookings in the backend
- **Operating Environment**
  - Client: Modern web browsers (Chrome, Firefox, Safari)
  - Server: Web backend (e.g., Node.js, Python) with relational or NoSQL database
- **Assumptions and Dependencies**
  - Reliable internet connectivity for users
  - Availability of a payment gateway API (e.g., Stripe, PayPal)

5. Visual Mockup Reference

Design wireframes/screenshots should include the following pages:

- Landing/Home
- Login/Sign-up
- Search Results
- Flight Selection/Details
- Book a Flight
- Booking Confirmation
- My Bookings
- User Profile/Account

Live design available on Figma:
Airline Booking System UI/UX (Figma)
(https://www.figma.com/make/2TIfAaviZLvRTfY95KbdWj/Airline-Booking-System-UI-UX?node-id=0-4&t=rJMVaOfTQaEewtvx-0)

6. Features

The Airline Booking System is structured around the following core features:

- **Landing/Home**: Provides the initial entry point and search interface.
- **User Registration and Login**: Allows users to create accounts and log in securely.
- **Flight Search & Results**: Allows users to find and view available flights by origin, destination, and date.
- **Flight Selection/Details**: Allows users to choose a specific flight and view full details before booking.
- **Booking Creation (Input Forms)**: Users can select flights, provide passenger details, and finalize the booking inputs.
- **Payment & Booking Confirmation**: Secure online payment integration to complete the booking, leading to a confirmation.
- **Ancillary Services Purchase**: **Purchase additional services (meals, baggage, shop items)** during the booking process or post-booking.
- **Booking Management (My Bookings)**: Users can view and manage existing bookings.
- **User Profile/Account**: Allows users to manage personal information and security settings.

7. Functional Requirements

Use Cases (Core User Journeys)

- **Use Case 1: User Registration and Login**
  - **Description**: New users can register and existing ones can log in securely.
  - **Actors**: End User
  - **Preconditions**: User accesses registration or login page
  - **Postconditions**: User account created or user authenticated
  - **Main Flow**: User inputs email and password > System validates and registers/authenticates user > User accesses account dashboard
  - **Alternate Flow**: Invalid input leads to error messages
- **Use Case 2: Flight Search**
  - **Description**: User searches for flights by origin, destination, and date.
  - **Actors**: End User
  - **Preconditions**: User is on flight search page
  - **Postconditions**: System displays available flights matching criteria
  - **Main Flow**: User inputs search parameters > System queries flights database > Results displayed
  - **Alternate Flow**: No flights found message if no matches
- **Use Case 3: Booking a Flight and Add-ons**
  - **Description**: User books a selected flight, optionally adds ancillary services, and

provides payment.
    - **Actors**: End User
    - **Preconditions**: User is logged in and has selected a flight
    - **Postconditions**: Booking confirmed, ticket issued, and ancillary services purchased
    - **Main Flow**: User selects flight > enters passenger info > selects optional meals/baggage/shop items > makes payment > confirms booking
    - **Alternate Flow**: Payment failure triggers error and retry option

System Features (Detailed Requirements)

1. **User Authentication**
    - Users must be able to register and log in with valid credentials.
2. **Flight Search**
    - Users must be able to search for available flights based on origin, destination, and date.
3. **Booking Management**
    - Users can create, view, and cancel bookings.
4. **Payment Processing**
    - Users must be able to pay for tickets and add-ons through an integrated payment gateway.
5. **Add-on Services**
    - The system allows the purchase of meals, baggage, and in-flight shop items.
6. **Admin Management**
    - Admin users can add, edit, or delete flight schedules and manage bookings.

8. Non-Functional Requirements

- **Performance**: The system should handle multiple concurrent users without performance degradation. Search results should return within 3 seconds.
- **Security**: Sensitive information such as passwords and payment details must be encrypted. Passwords should be hashed, and HTTPS used for all transactions.
- **Availability**: The system should maintain a high level of uptime (target 99.5%).
- **Usability**: The interface should be intuitive and responsive across devices (desktop, tablet, mobile).
- **Scalability**: The system architecture should support expansion of services and features.
- **Maintainability**: Codebase should be modular and easy to update, with good documentation.

9. Data Requirements

- Entity-Relationship Diagram (ERD):
  The data model describes the relationships between core entities like User, Flight, Airport, Booking, Passenger, Payment, and Flight Seat.

- **Database Requirements**: NoSQL (e.g., MongoDB Atlas) or Relational (as implied by ERD) to store user, flight, and booking data efficiently.

10. External Interface Requirements

- User Interface (UI):
  Browser-based web interface accessible via modern web browsers, ensuring responsive design for all devices.
- Software Interfaces:
  Integration with third-party payment APIs such as PayPal or Stripe. Optional integration with external flight data APIs.
- Hardware Interfaces:
  Standard computing devices (desktop, laptop, or mobile).
- Communications Interfaces:
  Internet connection required for all user interactions and API calls.

11. Glossary

Term	Definition
PNR	Passenger Name Record, a unique identifier for a booking.
API	Application Programming Interface, allows systems to communicate.
ERD	Entity-Relationship Diagram, a visual representation of the database structure.
MVP	Minimum Viable Product, a version of a product with just enough features to satisfy early customers.

12. Appendices

- A. Project Management:
  Project Trello Board
  (https://trello.com/invite/b/6906e8ed05cce0fc4ce0efb5/ATTIbbadefc4fa581855acadfea67789fb0e506C8468/mcp-side-project-phase-i-flight-booking-system)
- B. Design Reference:
  Airline Booking System UI/UX on Figma

(https://www.figma.com/make/2TIfAaviZLvRTfY95KbdWj/Airline-Booking-System-UI-UX?node-id=0-4&t=rJMVaOfTQaEewtvx-0)

- C. Data Model Visualization:
  The complete ERD can be viewed here: Airline Booking System ERD