**Flight Booking System Documentation**

Welcome to the Flight Booking System documentation! This concise guide offers key insights into the system's features, architecture, API endpoints, and deployment guidelines. Get started with our advanced flight booking solution!

Table of Contents

[Introduction 1](#_Toc150301379)

[Features 2](#_Toc150301380)

[Architecture 2](#_Toc150301381)

[Getting Started 2](#_Toc150301382)

[Prerequisites 2](#_Toc150301383)

[Installation 2](#_Toc150301384)

[Build and Run 3](#_Toc150301385)

[Usage 3](#_Toc150301386)

[User Registration and Login 3](#_Toc150301387)

[Flight Search and Booking 3](#_Toc150301388)

[Promo Code Redemption 3](#_Toc150301389)

[Admin Dashboard 3](#_Toc150301390)

[Data Models 4](#_Toc150301391)

[Error Handling 4](#_Toc150301392)

[Roles and Permissions 5](#_Toc150301393)

[Entity-Relationship Diagram (ERD) Outline: 7](#_Toc150301394)

[Relationships: 8](#_Toc150301395)

[API Endpoints 8](#_Toc150301396)

[Request and Response Formats 9](#_Toc150301397)

[User Registration Request: 9](#_Toc150301398)

[User Login Request: 9](#_Toc150301399)

[Flight Search Request: 9](#_Toc150301400)

[Flight Booking Request: 9](#_Toc150301401)

[Response Format: 10](#_Toc150301402)

[**Error Response:** 10](#_Toc150301403)

[Flight SearchResponse**:** 11](#_Toc150301404)

[FlightBooking Response: 11](#_Toc150301405)

# 

# Introduction

The Flight Booking System is a robust web application designed to simplify flight booking processes. It enables users to search for flights, book tickets, apply promo codes for discounts, and manage bookings efficiently. The system includes a user-friendly interface for customers and a powerful admin dashboard for flight and promo code management.

# Features

* **User Management:**
  + User Registration
  + User Login and Authentication
* **Flight Booking:**
  + Flight Search by Airport Codes
  + Seat Selection
  + Ticket Booking Confirmation via Email
* **Promo Code Redemption:**
  + Promo Code Application for Discounts
* **Admin Dashboard:**
  + Flight Management (Add, Update, Delete)
  + Promo Code Management (Generate, Expire)

# Architecture

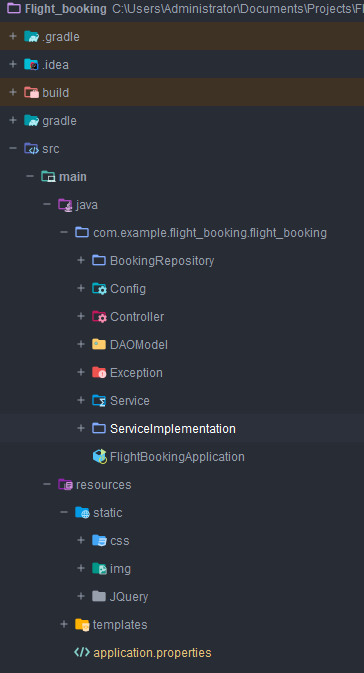
The Flight Booking System follows a layered architecture:

* **Presentation Layer:** Thymeleaf templates for UI rendering.
* **Controller Layer:** Handles incoming requests and orchestrates the flow.
* **Service Layer:** Business logic for flights, bookings, and promo codes.
* **Repository Layer:** MongoDB repositories for data storage.

# Project Structure

The Flight Booking System follows a well-organized structure, ensuring modularity and scalability. Here's an overview:

* **Controller:** Handles incoming requests and manages the flow of data between the Model and View components.
* **Service:** Implements business logic, processes data, and interacts with repositories.
* **Repository:** Manages data persistence and communicates with the database.
* **Config:** Contains configuration files for various components like security and email settings.
* **DAOModel:** Defines data access objects and entities used in the database.
* **Exception:** Houses custom exception classes for handling specific errors.
* **Utility:** Provides utility classes and helper functions for different modules.
* **Resources:** Stores static resources such as CSS files, logos, and JavaScript files.
* **Templates:** Contains HTML templates for rendering dynamic web pages.



# Getting Started

## Prerequisites

* **Java 17:** Ensure you have Java 17 installed on your system.
* **MongoDB:** Install and configure MongoDB on your local machine.
* **Gradle:** Build tool for the project.

# Installation

1. **Clone the Repository:**

git clone https://github.com/moxhadeel571/Flight\_Booking\_SpringBoot.git

**Configure MongoDB:**

Update MongoDB connection settings in src/main/resources/application.properties.

Build and Run**:**

shell

1. gradle build
2. java -jar build/libs/flight-booking.jar
3. Access the application at http://localhost:9090.

# Usage

## User Registration and Login

1. **Register:**
   * Visit the registration page and provide necessary details.
   * Choose user role (customer, admin).
2. **Login:**
   * Access the login page and enter your credentials.
   * Choose the appropriate role to access customer or admin features.

## Flight Search and Booking

1. **Search for Flights:**
   * Enter departure and arrival airport codes.
   * View available flights.
2. **Book a Ticket:**
   * Select a flight.
   * Choose seats and passengers.
   * Confirm booking.
3. **Ticket Confirmation:**
   * Receive an email confirmation with a PDF ticket.

## Promo Code Redemption

1. **Redeem Promo Code:**
   * During checkout, enter a valid promo code.
   * The discount will be applied to the total amount.

## Admin Dashboard

1. **Admin Login:**
   * Use admin credentials to access the admin dashboard.
2. **Manage Flights:**
   * Add new flights with details.
   * Update existing flight information.
   * Delete flights.
3. **Manage Promo Codes:**
   * Generate new promo codes with specific discounts.
   * Expire existing promo codes.

# Data Models

The system utilizes the following data models:

* **User:**
  + id: Unique identifier
  + username: User's username
  + password: User's encrypted password
  + email: User's email address
  + roles: User roles (customer, admin)
* **Flight:**
  + id: Unique identifier
  + departureAirportCode: Departure airport code
  + arrivalAirportCode: Arrival airport code
  + price: Ticket price
  + availableSeats: Number of available seats
* **Booking:**
  + id: Unique identifier
  + userId: ID of the booking user
  + flightId: ID of the booked flight
  + passengers: List of passenger details
  + totalPrice: Total booking price
* **Promo Code:**
  + id: Unique identifier
  + code: Promo code string
  + discountPercentage: Discount percentage

# 

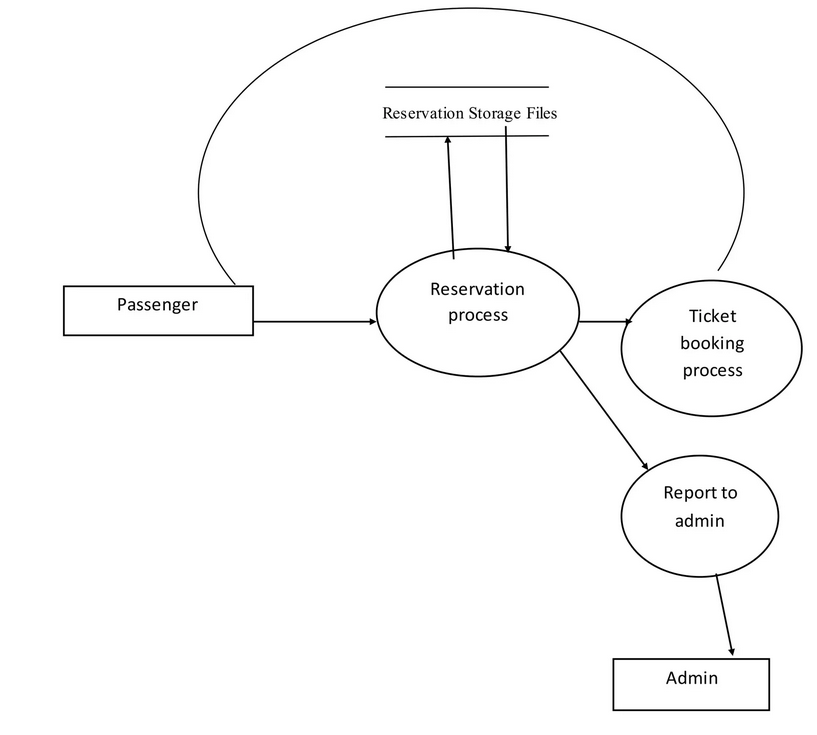
# Error Handling

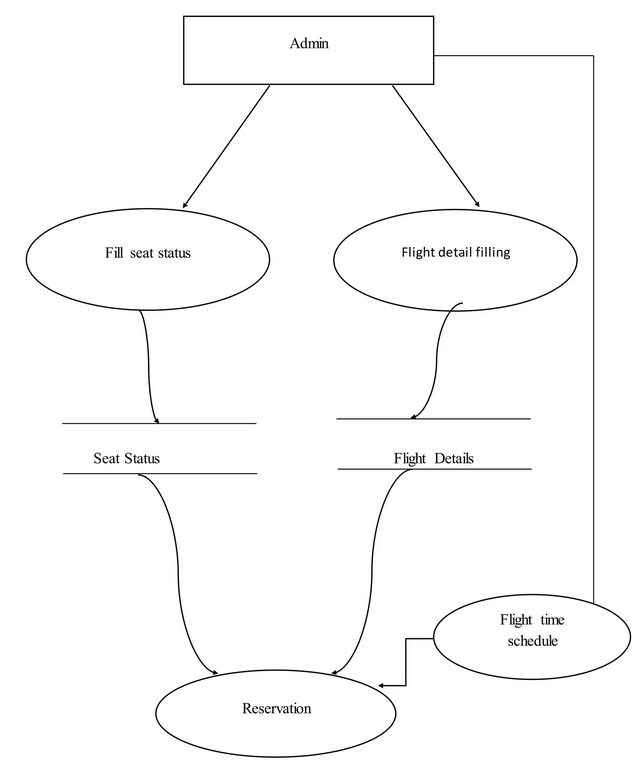
The Flight Booking System handles errors gracefully, providing meaningful error responses in JSON format. Error responses include an error message and an appropriate HTTP status code to indicate the nature of the error (e.g., 400 for bad request, 401 for unauthorized access, 500 for internal server error).

# Roles and Permissions

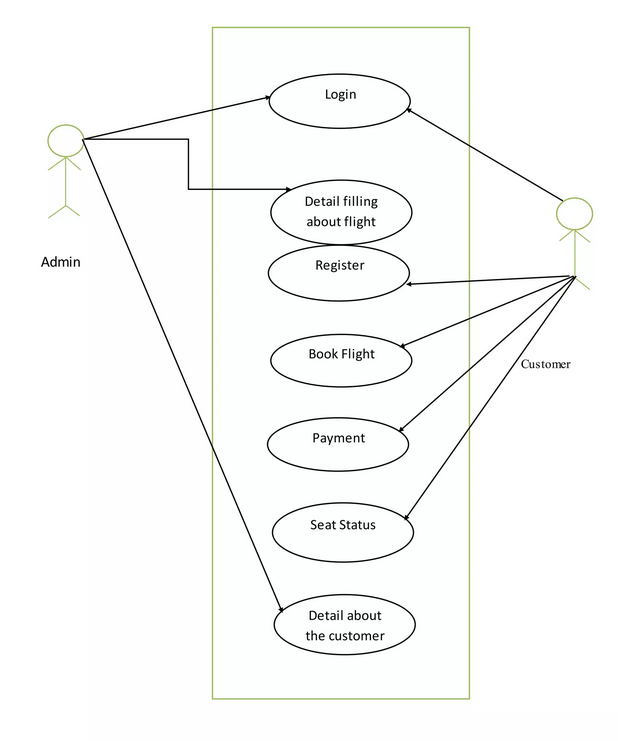
The Flight Booking System implements role-based access control to restrict user actions and access to specific features. The following roles are defined in the system:

* **ADMIN**: Users with the ADMIN role have access to airline-related functionalities, including managing flights, viewing the dashboard, and generating coupons.





* **USERS**: Users with the USERS role can search for flights, make bookings, apply promo codes, and download PDF tickets.



Roles are assigned during user registration or account creation. The system checks the user's role before allowing access to certain endpoints or features.

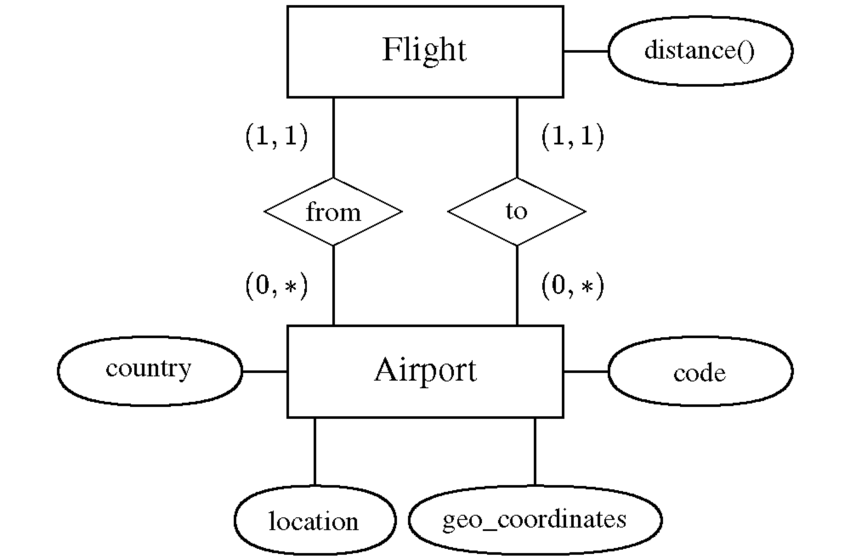
# 

# Entity-Relationship Diagram (ERD) Outline:

* **User:** UserID (PK), Username, Password, Email, Role
* **Flight:** FlightID (PK), FlightNumber, DepartureAirport, ArrivalAirport, DepartureDateTime, ArrivalDateTime, Price, AvailableSeats
* **Booking:** BookingID (PK), UserID (FK), FlightID (FK), BookingDateTime, TotalPrice
* **Passenger:** PassengerID (PK), BookingID (FK), Name, Email, SeatNumber
* **Coupon:** CouponID (PK), CouponCode, DiscountPercentage

## Relationships:

* **User-Booking:** One-to-Many
* **Flight-Booking:** One-to-Many
* **Booking-Passenger:** One-to-Many
* **User-Role:** Many-to-Many
* **Booking-Coupon:** One-to-One



# API Endpoints

The Flight Booking System provides the following API endpoints:

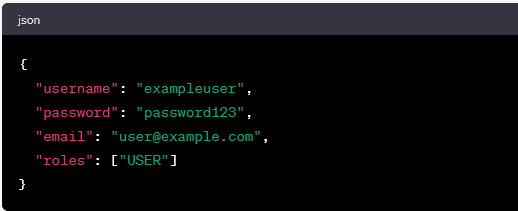
* **POST /api/register:** Register a new user.
* **POST /api/login:** User login and authentication.
* **GET /api/flights:** Get a list of available flights.
* **POST /api/bookings:** Create a new booking.
* **POST /api/promo-codes:** Apply a promo code to a booking.

# Request and Response Formats

**Request Format:**

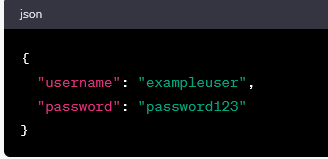
## User Registration Request:

* + Endpoint: POST /register
  + Body:



## User Login Request:

* Endpoint: POST /signin
* Body:



# Flight Search Request:

* Endpoint: GET /customer/search
* Parameters:
  + airport\_from: "Origin Airport Code"
  + airport\_to: "Destination Airport Code"

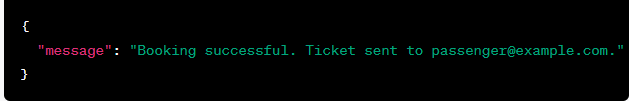
## Flight Booking Request:

* Endpoint: POST /customers/booking/{id}
* Path Variable: id (FlightID



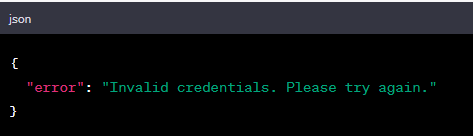
## Response Format:

1. **Success Response:**
   * Status: 200 OK
   * Body (Example):



## **Error Response:**

* Status: 4xx, 5xx (Depending on the type of error)
* Body (Example):



## 

## Flight SearchResponse**:**

* Status: 200 OK
* Body (Example):



## FlightBooking Response:

* Status: 200 OK
* Body (Example):

