**TASK – 4 AIRLINE RESERVATION SYSTEM**

# Airline Reservation System Documentation

## Table of Contents

1. [Introduction](#introduction)

- [Purpose](#purpose)

- [Scope](#scope)

- [Features](#features)

2. [Architecture](#architecture)

- [Backend](#backend)

- [Database](#database)

3. [System Components](#system-components)

- [Entities](#entities)

- [Services](#services)

- [Controllers](#controllers)

4. [How to Use](#how-to-use)

- [Prerequisites](#prerequisites)

- [Installation](#installation)

- [Login](#login)

- [Ticket Booking](#ticket-booking)

5. [Development](#development)

- [Technologies Used](#technologies-used)

- [Project Structure](#project-structure)

- [Testing](#testing)

6. [Contributing](#contributing)

7. [License](#license)

## Introduction

### Purpose

The Airline Reservation System is a Java-based application designed to provide a user-friendly platform for booking and managing airline tickets. It facilitates the scheduling of online appointments, allows doctors to provide medical services, and connects potential organ and blood donors.

### Scope

The system includes two key modules: Admin and Doctor. Admins can manage users, appointments, and system configurations, while doctors can access patient records and provide medical services.

### Features

- User authentication for admins and doctors.

- Appointment scheduling for patients.

- Doctor access to patient records.

- Connection with potential organ and blood donors.

## Architecture

### Backend

The backend of the Airline Reservation System is built using Java with the Spring Boot framework. It includes entity classes representing passengers, doctors, and appointments, as well as services for managing these entities.

### Database

A relational database management system (RDBMS) is used to store data related to passengers, doctors, and appointments. The database design ensures proper relationships between entities.

## System Components

### Entities

#### Passenger

- Represents a passenger booking a flight.

- Includes attributes such as first name, last name, and ticket type.

#### Doctor

- Represents a doctor providing medical services.

- Includes attributes such as name.

#### Appointment

- Represents a scheduled appointment between a patient and a doctor.

- Includes attributes such as appointment date and time.

### Services

#### AuthenticationService

- Provides methods for user authentication.

#### TicketBookingService

- Provides methods for booking flight tickets.

### Controllers

#### LoginPage

- Handles user authentication and redirects to the ticket booking page upon successful login.

#### TicketBookingPage

- Displays ticket booking options and handles the booking process.

## How to Use

### Prerequisites

- Java Development Kit (JDK) installed

- Integrated Development Environment (IDE) like IntelliJ IDEA or Eclipse

### Installation

1. Clone the repository to your local machine:

git clone <https://github.com/>uggapallysagar/airline.git

2. Open the project in your preferred IDE.

### Login

1. Open the terminal in your IDE or command prompt.

2. Navigate to the project directory:

```bash

cd path/to/airline-reservation-system

```

3. Run the login page:

```bash

java LoginPage.java

```

4. Enter the following credentials:

- Username: `admin`

- Password: `admin`

5. Upon successful login, you will be directed to the ticket booking options.

### Ticket Booking

1. After logging in, you will see the available ticket booking options:

- Domestic

- International

2. Choose the appropriate option by entering the corresponding number.

3. Enter passenger details as prompted.

4. The system will confirm the successful ticket booking.

## Development

### Technologies Used

- Java

- Spring Boot

- RDBMS (MySQL, PostgreSQL, etc.)

### Project Structure

The project follows a modular structure:

- \*\*Model:\*\* Contains entity classes.

- \*\*Service:\*\* Contains service classes for business logic.

- \*\*UI:\*\* Contains user interface classes (controllers) for handling user interactions.

- \*\*Main Application:\*\* Contains the main class to run the application.

### Testing

- Unit testing, integration testing, and end-to-end testing are conducted to ensure the reliability of the system.

## Contributing

If you'd like to contribute to the development of this project, please follow the guidelines outlined in the [CONTRIBUTING](CONTRIBUTING.md) file.

###REQUEST YOU TO CHECK OUT CODE FROM GITHUB CODE FILES SINCE IT HA A LOT OF FILE I CANT INCLUDE CODE HERE

OUTPUT IMAGES :

  
  
  
