**FLIGHT RESERVATION SYSTEM**

**Context About the Topic:**

Here we devise a system which helps a user to search and book flights as per one’s requirement.

* Customer will land at the Home Page and will be able to search flights and view information as per their need (even without signing-up).
* Customer need to sign-up(register) and enter their details if they want to book any flight, or they need to login if previously signed-up.
* Admin has dedicated services with which admin maintains the information to be displayed.

Below given are the section wise information of further workings -

**BACKEND:**

**Database Type:** MongoDB

**Database Name:** FMS (demo\_fms)

It will store various collections such as:

**[1] Flights**:

**->** Flight ID [auto-generated from system]: **Long**

**->** Flight No. [will be unique]: **String**

**->** Carrier: **String**

**->** Seating Capacity: **Long**

**->** Source Airport-ID: **Long**

**->** Destination Airport-ID: **Long**

**[2] Users:**

**->** User ID [auto-generated from system] [Unique]: **Long**

**->** First Name: **String**

**->** Last Name: **String**

**->** Email [Unique]: **String**

**->** Password: **String**

**->** User Type: **String**

**[3] Airports:**

**->** Airport ID [auto-generated from system] [Unique]: **Long**

**->** Airport Code [Unique]: **String**

**->** Airport Name [Unique]: **String**

**->** Airport Location: **String**

**[4] Bookings:**

**->** Booking ID [auto-generated from system] [Unique]: **Long**

**->** Unique Code: **String**

**->** Booking Date: **String**

**->** User ID: **Long**

**->** No of Passengers: **Long**

**->** Flight ID: **Long**

**[5] Database Sequences:**

**->** Sequence Id for all other document Id fields: **Long**

**MIDDLE-WARE:**

**Implemented Using:** Spring Boot

**Microservices:** 8

**Writing Style:** Camel Case

**[1] LOGIN-SERVICE:**

Server Port: 8182

Application Name: LOGIN-SERVICE

Base-Url: http://localhost:8182/login

API-Endpoints:

1. Create User (POST):

Validates User Credentials from Database.

**[2] USER-SERVICE:**

Server Port: 8081

Application Name: USER-SERVICE

Base-Url: http://localhost:8081/users

API-Endpoints:

1. Create User (POST): /create-user
2. Update User (PUT): /update-user/{userId}
3. Get All Users (GET): /get-all-users
4. Delete User (DELETE): /delete-user/{userId}
5. Search User (GET): /search-user/{userId}
6. Search Users by Name (GET):

/search-user-by-firstName/{fName}

1. Search Users by User Type (GET):

/search-user-by-type/{userType}

**[3] AIRPORT-SERVICE:**

Server Port: 8082

Application Name: AIRPORT-SERVICE

Base-Url: http://localhost:8082/airports

API-Endpoints:

1. Create Airport (POST): /create-airport
2. Update Airport (PUT): /update-airport/{airportId}
3. Get All Airports (GET): /get-all-airports
4. Delete Airport (DELETE): /delete-airport/{airportId}
5. Search Airport (GET): /search-airport/{airportId}
6. Search Airport by Name (GET):

/search-airport-by-name/{airportName}

1. Search Airports by Location (GET):

/search-airport-by-loc/{airportLocation}

1. Airport Search by Location (for restTemplate) (GET):

/airport-search-by-loc/{airportLoc}

**[4] FLIGHT-SERVICE:**

Server Port: 8083

Application Name: USER-SERVICE

Base-Url: http://localhost:8083/flights

API-Endpoints:

1. Create Flight (POST): /create-flight
2. Update Flight (PUT): /update-flight/{flightId}
3. Get All Flights (GET): /get-all-flights
4. Delete Flight (DELETE): /delete-flight/{flightId}
5. Search Flight (GET): /search-flight/{flightId}
6. Search Flights by Carrier (GET):

/search-flight-by-carrier/{carrier}

1. Get Flight Route Info (GET): /flight-route/{flightId}
2. Flight Search (for restTemplate) (POST):

/flight-search

**[5] BOOKING-SERVICE:**

Server Port: 8084

Application Name: BOOKING-SERVICE

Base-Url: http://localhost:8084/bookings

API-Endpoints:

1. Create Booking (POST): /create-booking
2. Update Booking (PUT): /update-booking/{bookingId}
3. Get All Booking (GET): /get-all-bookings
4. Delete Booking (DELETE):

/delete-booking/{bookingId}

1. Search Booking (GET): /search-booking/{bookingId}
2. Search Bookings by User-ID (GET):

/get-bookings-by-userId/{userId}

1. Get Booking Description (GET):

/get-booking-desc/{bookingId}

**[6] SERVICE-REGISTRY:**

Server Port: 8761

Application Name: SERVICE-REGISTRY

Url: http://localhost:8761

Implemented Using: Eureka

Task: Registers every mentioned service with EUREKA Server, for assisting in Service Discovery

**[7] API-GATEWAY:**

Server Port: 8180

Application Name: API-GATEWAY

Base-Url: http://localhost:8180/

Task: Provides a unified single point of entry for all services

Online-Check for Hystrix-Dashboard:

http://localhost:8180/actuator/hystrix.stream

Fall-Back API-Endpoints implementing Circuit Breaker:

1. LOGIN-SERVICE: /loginServiceFallBack
2. USER-SERVICE: /userServiceFallBack
3. AIRPORT-SERVICE: /airportServiceFallBack
4. FLIGHT-SERVICE: /flightServiceFallBack
5. BOOKING-SERVICE: /bookingServiceFallBack

**[8] HYSTRIX-DASHBOARD:**

Server Port: 8181

Application Name: HYSTRIX-DASHBOARD

Base-Url: http://localhost:8181/hystrix

Task: Provides graphical real-time monitoring of services

**Goals Achieved:**

1. Logging
2. Exception Handling
3. Comments Written
4. Service Discovery using Eureka
5. API Gateway
6. Circuit Breakers
7. Hystrix Dashboard
8. Swagger API Docs
9. Test Cases for Service and Repository Classes

**FRONT END:**

This has below mentioned elements:

**[1] Home Page:**

-> containing all the elements such as links to other pages like About-Us, Contact, etc..

-> providing login/signup form functionality

-> flight search option

-> footer element (social media,etc.)

**[2] Login Page:**

-> Login functionality:

-> redirecting to user/admin page as per credentials

**[3] Register Page:**

-> Signup functionality

-> signing-up a user with provided details

**[4] User Page:**

-> View Details

-> Update Details

-> Book Flight

-> View Booked Flights

**[5] Admin Page:**

-> USER-SERVICE:

[a] View all Users [b] Add User

[c] Search Users by Criteria [d] Update User Details

[e] Delete User [f] View User Details

-> AIRPORT-AERVICE:

[a] View all Airports [b] Add Airport

[c] Search Airports by Criteria [d] Update Airport Details

[e] Delete Airport [f] View Airport Details

-> FLGHT-SERVICE:

[a] View all Flights [b] Add Flight

[c] Search Flights by Criteria [d] Update Flight Details

[e] Delete Flight [f] View Flight Details

[g] View Flight Route Information

-> BOOKING-SERVICE:

[a] View all Bookings [b] Cancel Booking

[c] Search Bookings by Criteria [d] Update Booking Detail

[e] View Booking Details

[f] View Full Booking Description Information

**[6] Contact Us Page:**

-> demo Contact Us Page with a form and an embedded map

**[7] About Us Page:**

-> demo About Us Page with some information