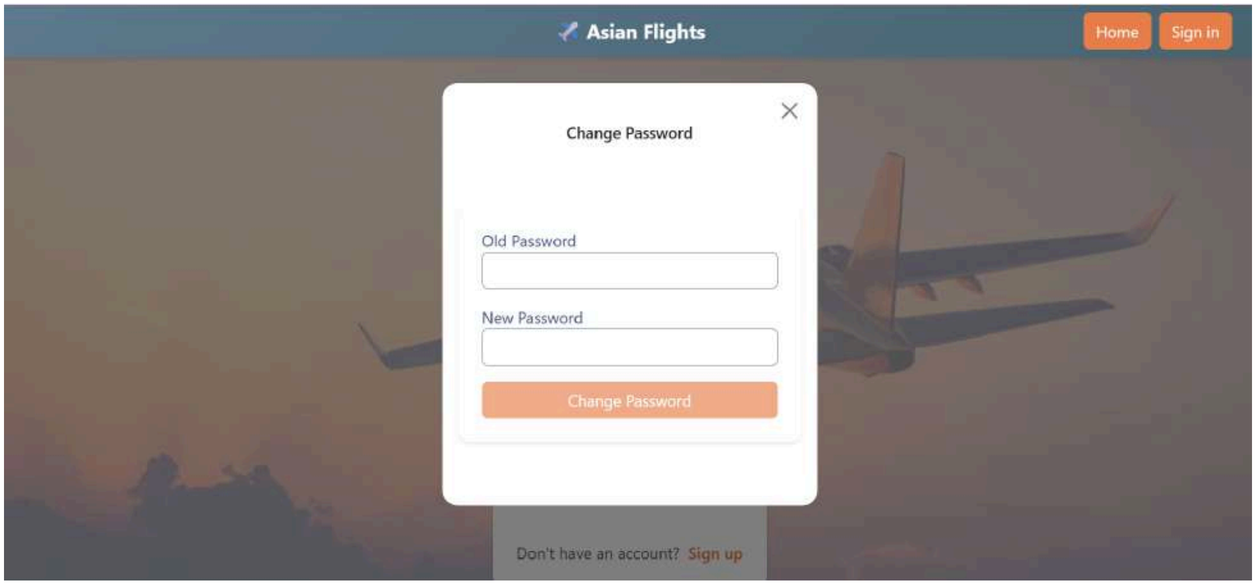
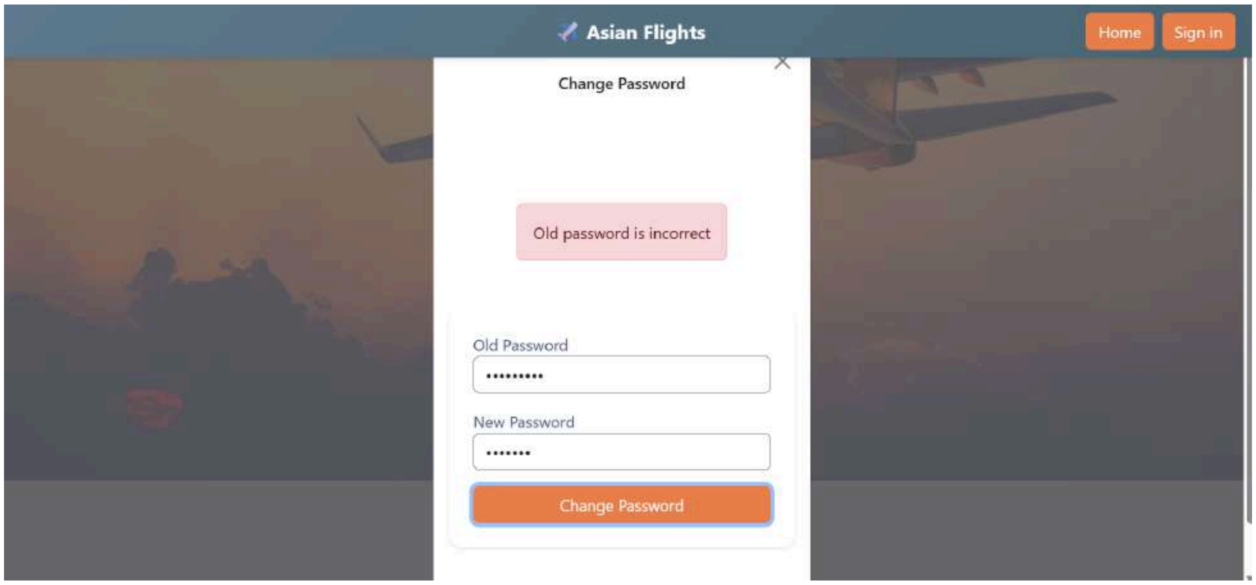
Repositories links:

- Frontend: https://github.com/asritha26k/frontend-for-flight-booking-app
- Backend: https://github.com/asritha26k/backend-flight-booking-app

Change password initiated after every 90 days:

(Tested with 15min change)



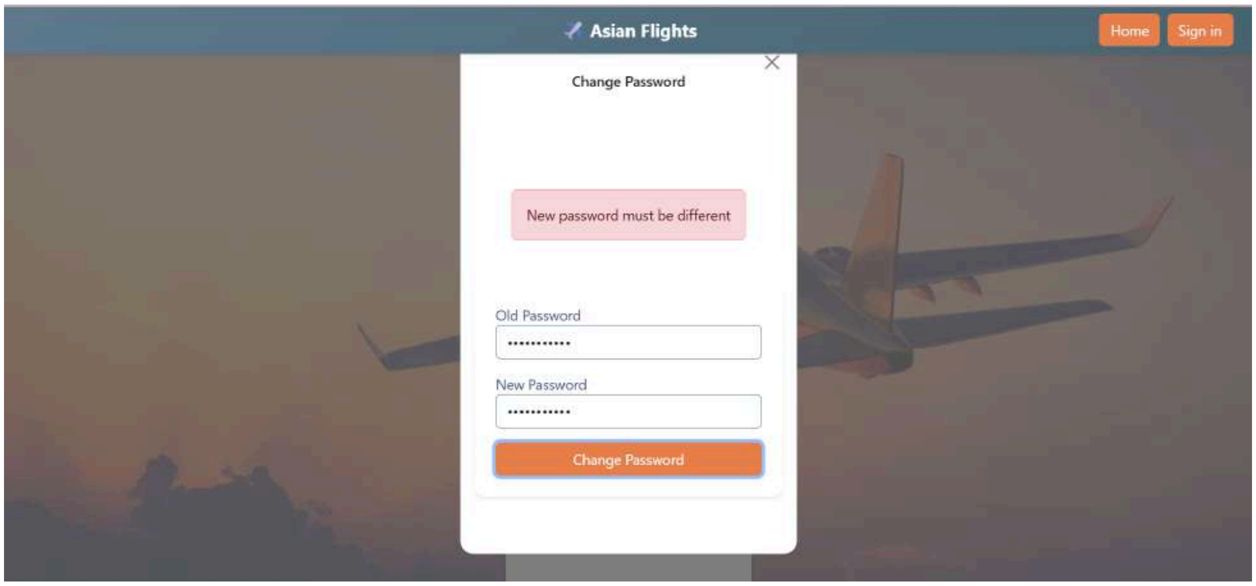


Change Password

Old password is incorrect

Old Password

New Password

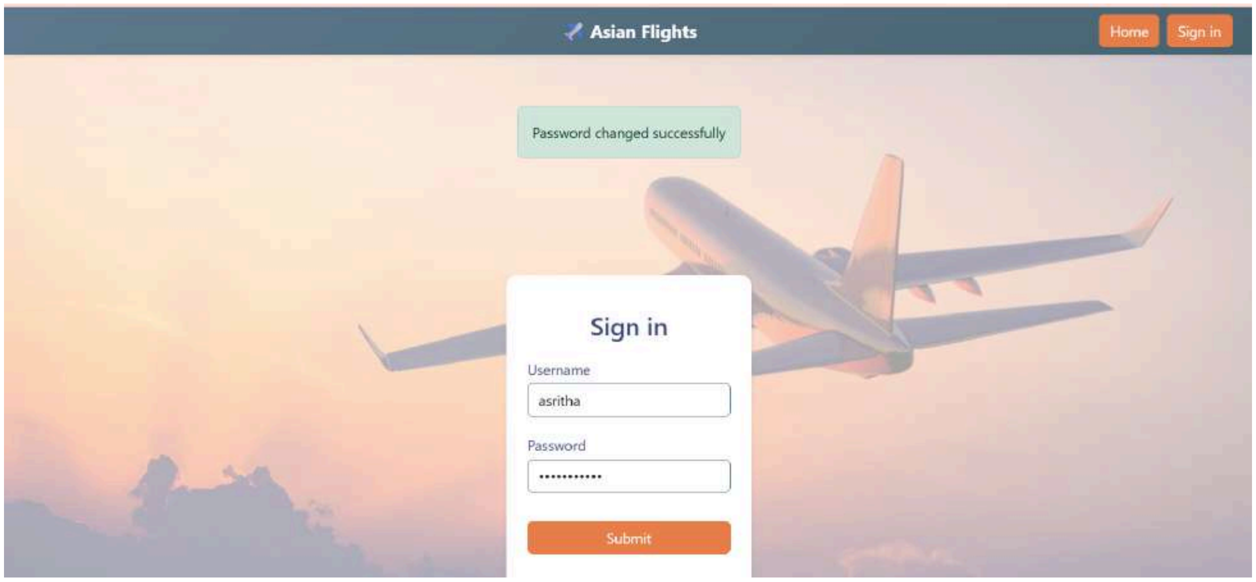


Change Password

New password must be different

Old Password

New Password



Password changed successfully

Sign in

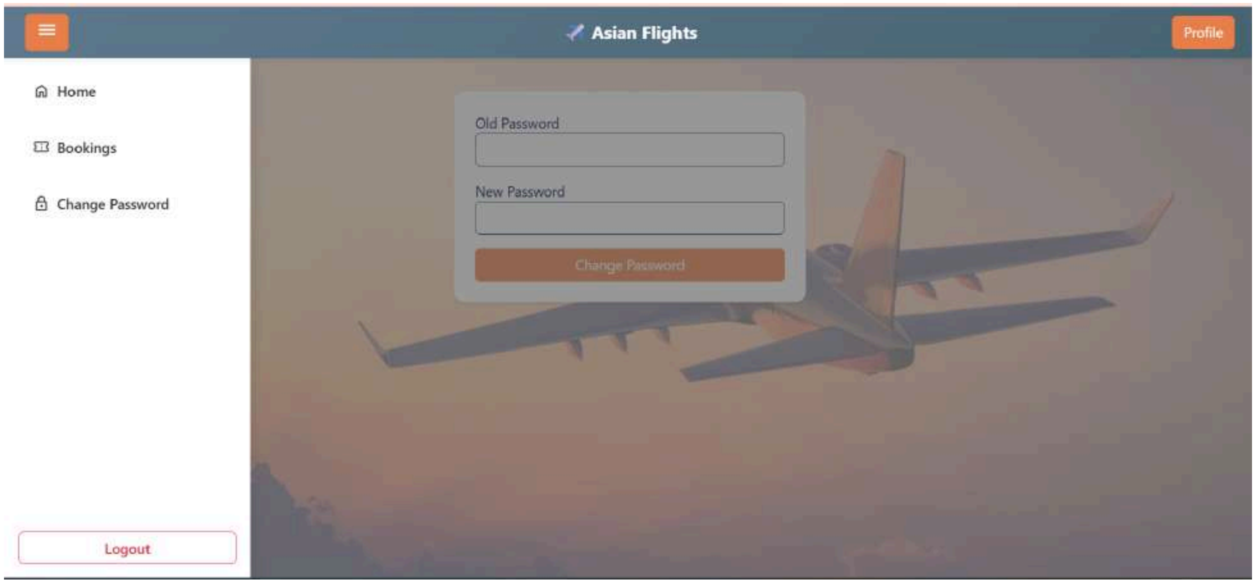
Username
asritha

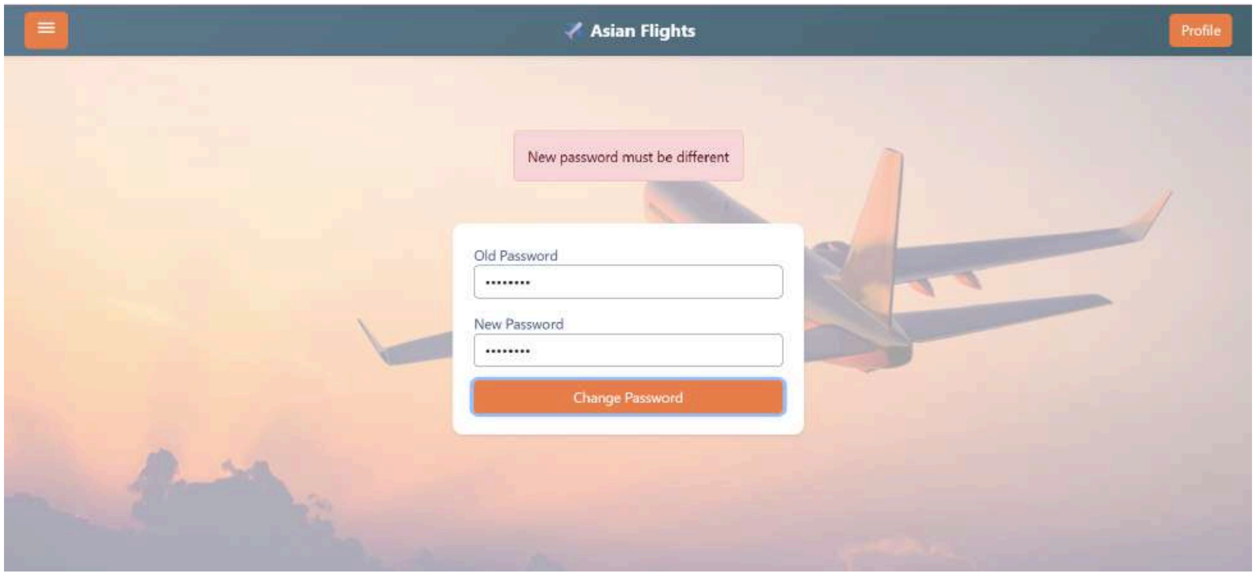
Password
•••••••••••

Change password feature if user wishes to change:

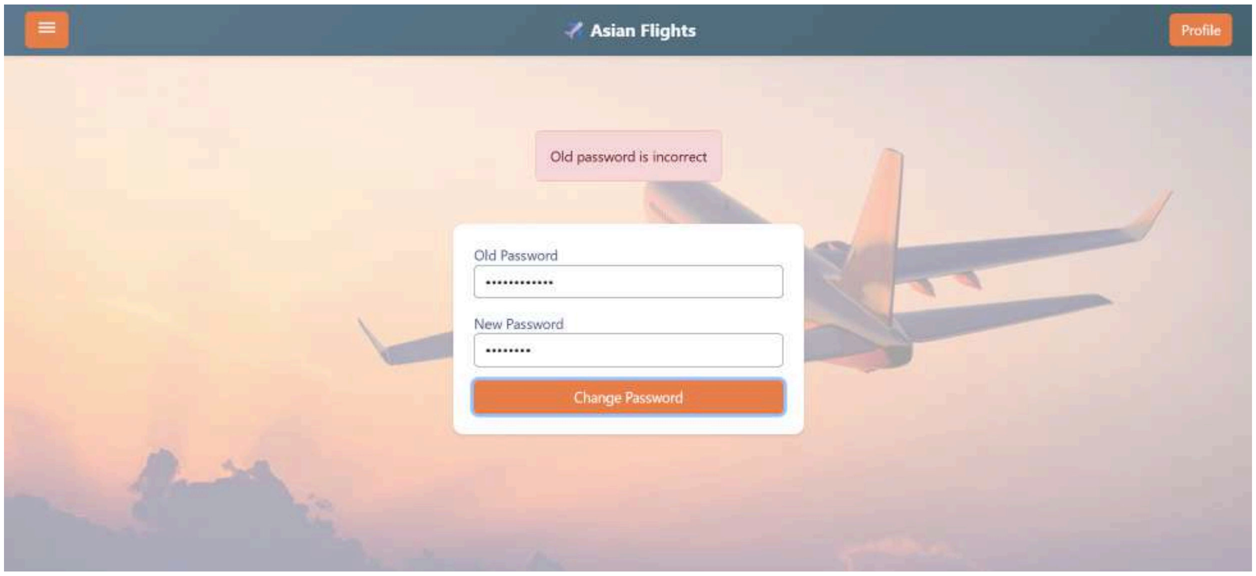
This is for admin role and user role both

After sign in



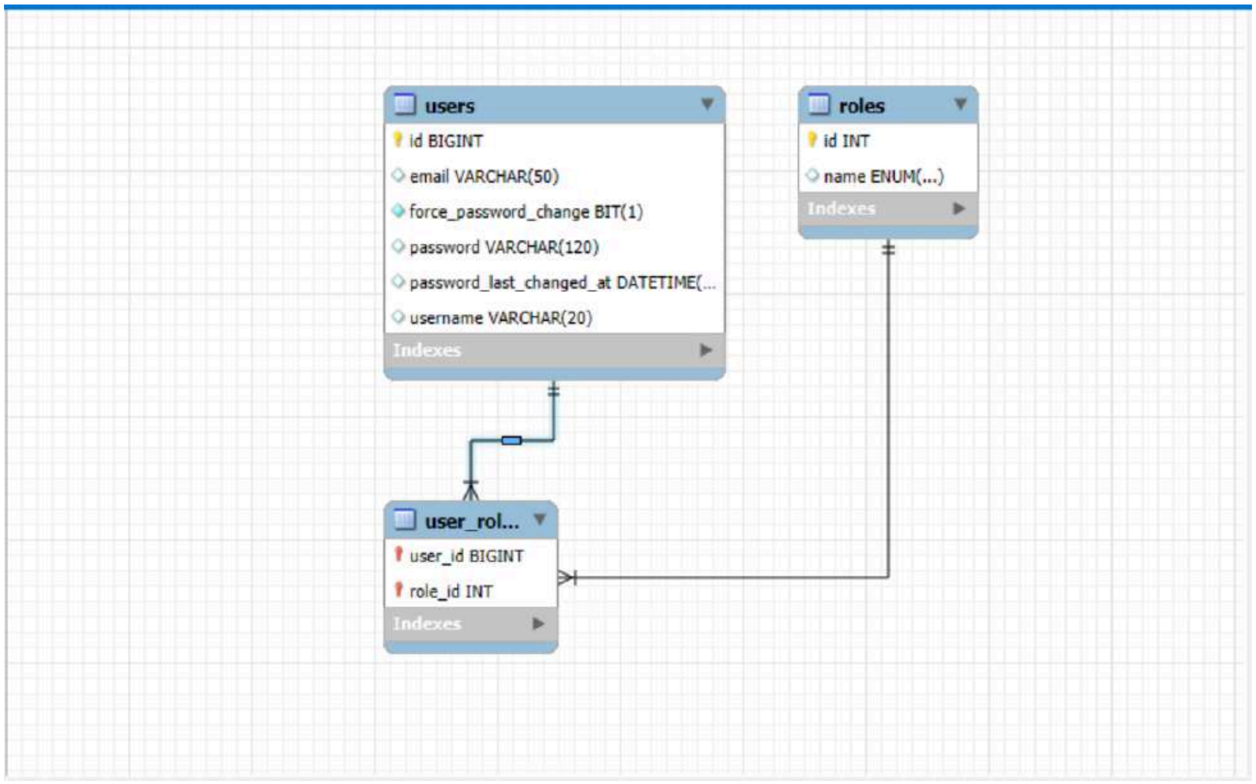


New password must be different



Old password is incorrect

Schema changes:



New api:

http://localhost:8765/auth-service/api/auth/change-password
Password changed successfully (200 ok)
Error: Old password is incorrect (400 Bad Request)

http://localhost:8765/auth-service/api/auth/signin
Sign in api:
After 90 days response

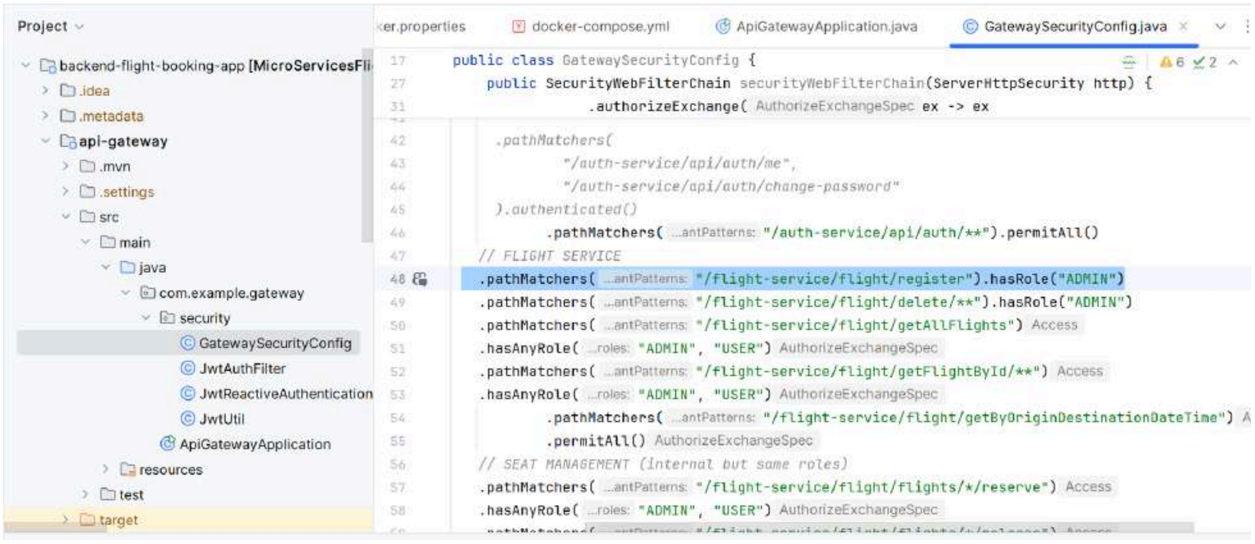
```
{
   "status": "PASSWORD_EXPIRED",
   "message": "Please change your password",
   "forcePasswordChange": true
}
```

Admin:

Role based access control

```
.pathMatchers("/flight-service/flight/register").hasRole("ADMIN")
```

In API GATEWAY
Backend handling:



```
public class GatewaySecurityConfig {
    public SecurityWebFilterChain securityWebFilterChain(ServerHttpSecurity http) {
                .authorizeExchange( ex -> ex
        .pathMatchers(
                "/auth-service/api/auth/me",
                "/auth-service/api/auth/change-password"
        ).authenticated()
                .pathMatchers( "/auth-service/api/auth/**").permitAll()
    // FLIGHT SERVICE
    .pathMatchers( "/flight-service/flight/register").hasRole("ADMIN")
    .pathMatchers( "/flight-service/flight/delete/**").hasRole("ADMIN")
    .pathMatchers( "/flight-service/flight/getAllFlights")
    .hasAnyRole( "ADMIN", "USER")
    .pathMatchers( "/flight-service/flight/getFlightById/**")
    .hasAnyRole( "ADMIN", "USER")
            .pathMatchers( "/flight-service/flight/getByOriginDestinationDateTime")
            .permitAll()
    // SEAT MANAGEMENT (internal but same roles)
    .pathMatchers( "/flight-service/flight/flights/*/reserve")
    .hasAnyRole( "ADMIN", "USER")
```

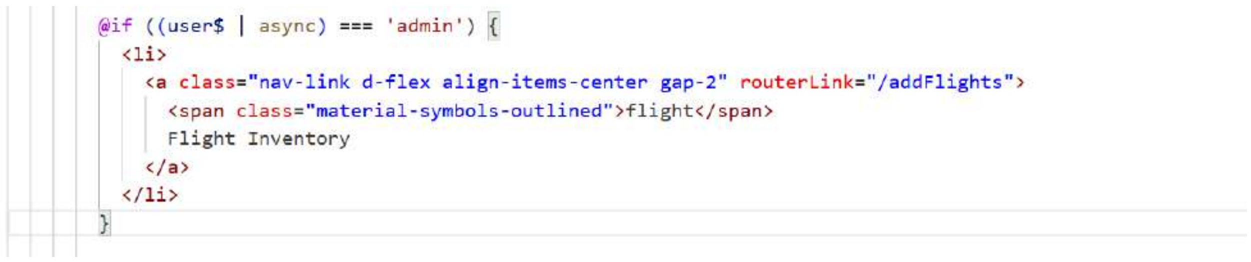
Frontend handling:

With the api:

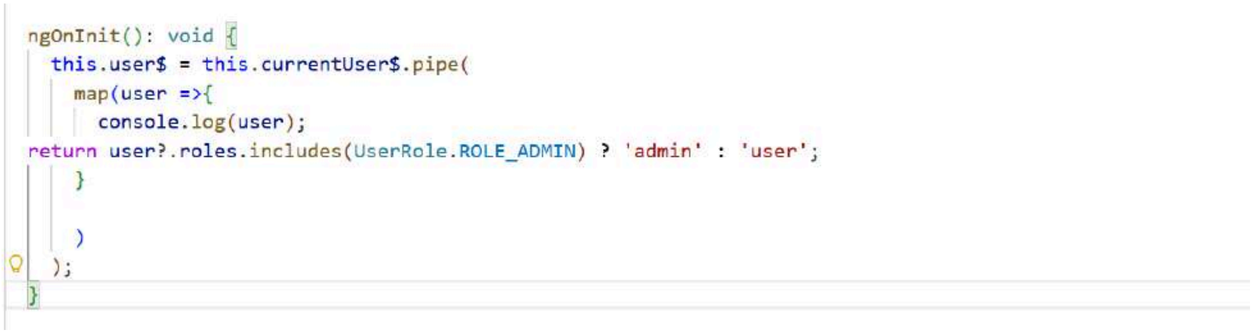
/auth-service/api/auth/me

Response:

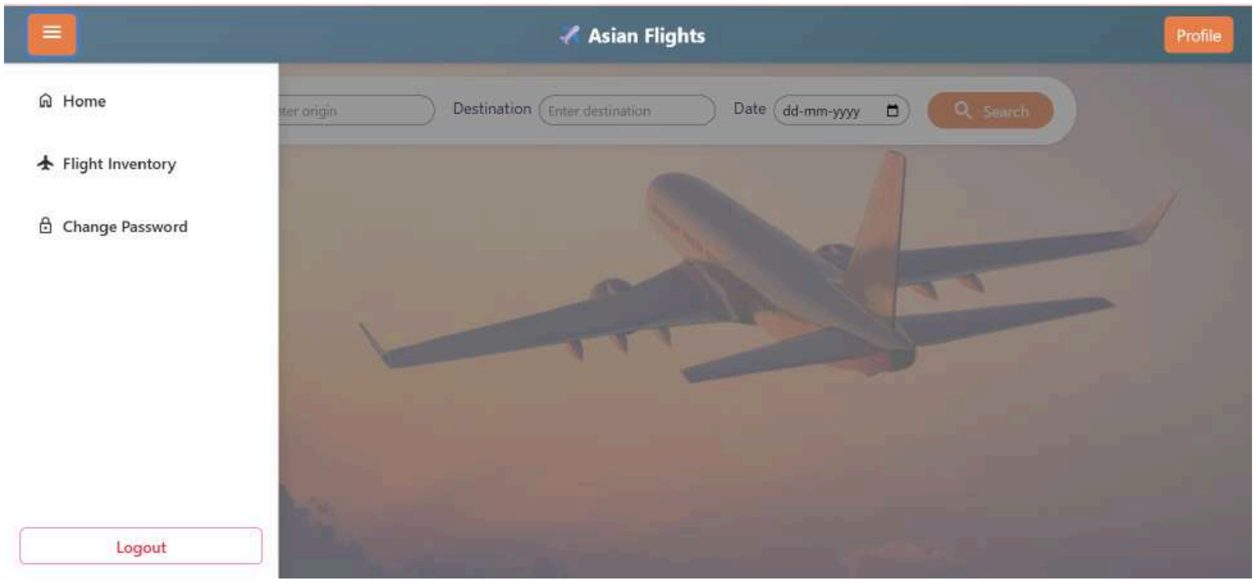
```
{
    "id": 5,
    "username": "admin",
    "email": "admin@gmail.com",
    "roles": [
        "ROLE_ADMIN"
    ]
}
```

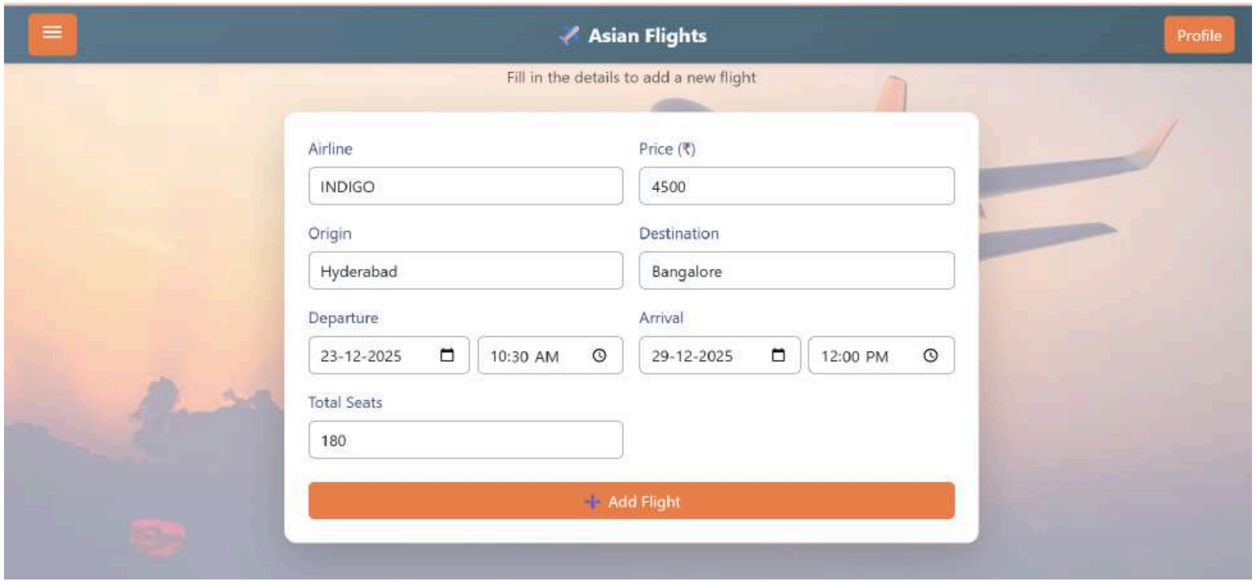


```
@if ((user$ | async) === 'admin') {
  <li>
    <a class="nav-link d-flex align-items-center gap-2" routerLink="/addFlights">
      <span class="material-symbols-outlined">flight</span>
      Flight Inventory
    </a>
  </li>
}
```



```
ngOnInit(): void {
  this.user$ = this.currentUser$.pipe(
    map(user =>{
      console.log(user);
return user?.roles.includes(UserRole.ROLE_ADMIN) ? 'admin' : 'user';
    }

    )
  );
}
```

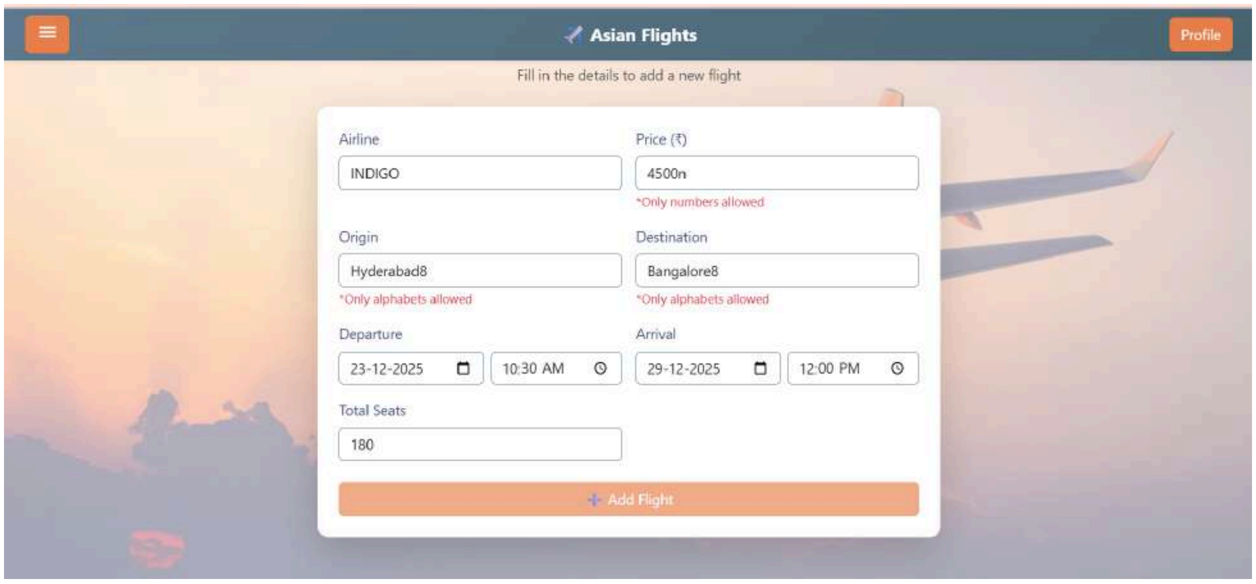




Fill in the details to add a new flight

Airline: INDIGO	Price (₹): 4500
Origin: Hyderabad	Destination: Bangalore
Departure: 23-12-2025 10:30 AM	Arrival: 29-12-2025 12:00 PM
Total Seats: 180	

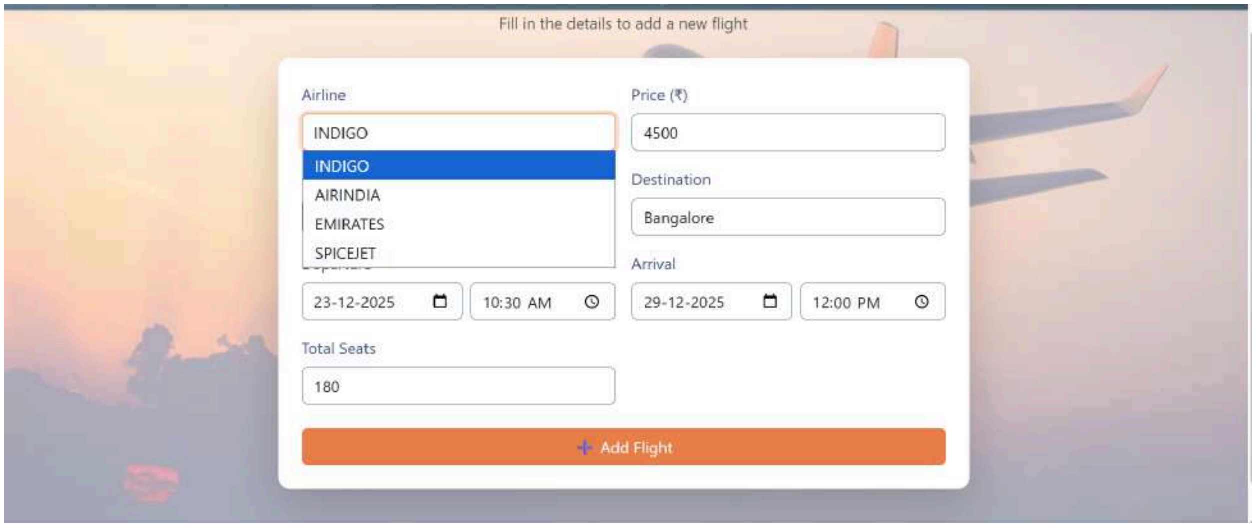
Validations:



Fill in the details to add a new flight

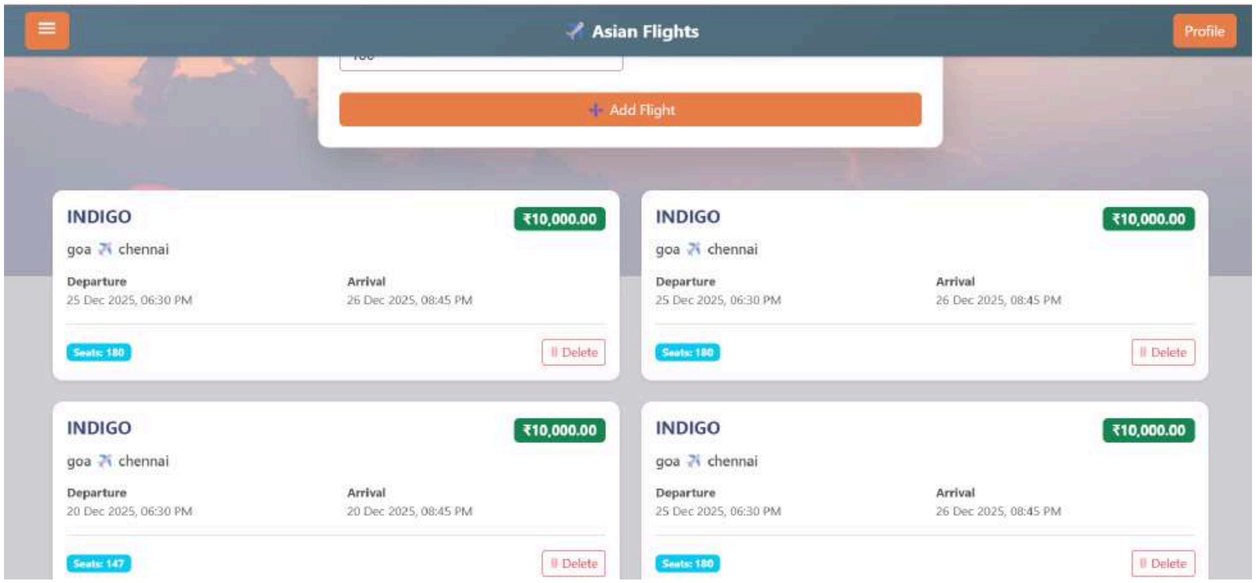
Airline: INDIGO	Price (₹): 4500n *Only numbers allowed
Origin: Hyderabad8 *Only alphabets allowed	Destination: Bangalore8 *Only alphabets allowed
Departure: 23-12-2025 10:30 AM	Arrival: 29-12-2025 12:00 PM
Total Seats: 180	

Drop down for Different airlines selection


Fill in the details to add a new flight

Airline: INDIGO	Price (₹): 4500
	Destination: Bangalore
Departure: 23-12-2025 10:30 AM	Arrival: 29-12-2025 12:00 PM
Total Seats: 180	

Flights shown after adding.



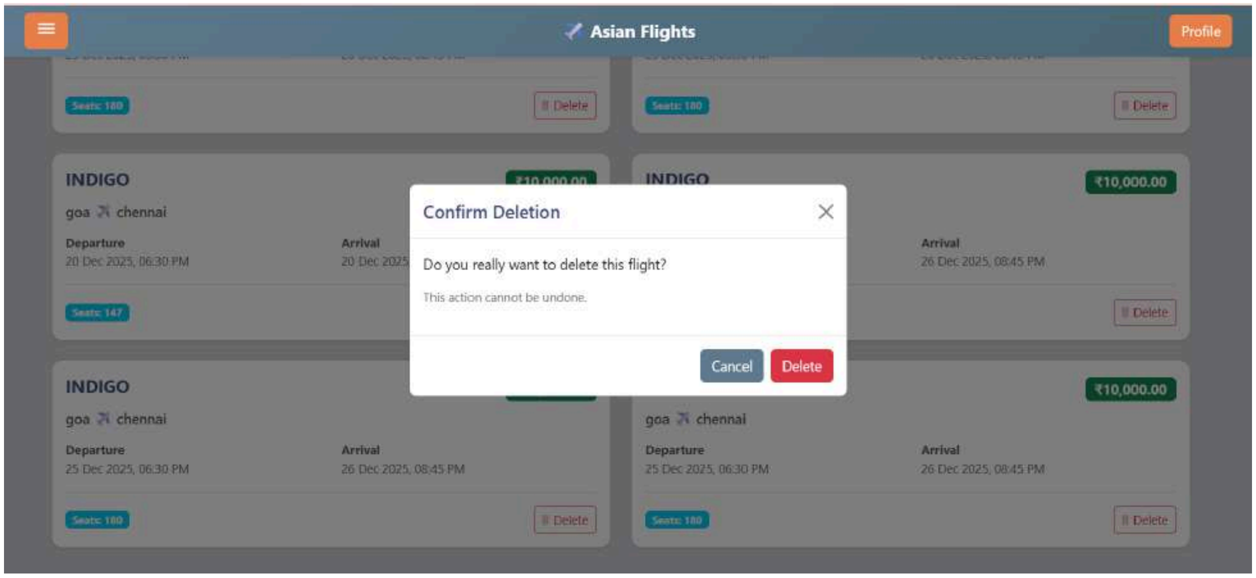
INDIGO ₹10,000.00
goa chennai
Departure 25 Dec 2025, 06:30 PM
Arrival 26 Dec 2025, 08:45 PM
Seats: 180 Delete

INDIGO ₹10,000.00
goa chennai
Departure 25 Dec 2025, 06:30 PM
Arrival 26 Dec 2025, 08:45 PM
Seats: 180 Delete

INDIGO ₹10,000.00
goa chennai
Departure 20 Dec 2025, 06:30 PM
Arrival 20 Dec 2025, 08:45 PM
Seats: 147 Delete

INDIGO ₹10,000.00
goa chennai
Departure 25 Dec 2025, 06:30 PM
Arrival 26 Dec 2025, 08:45 PM
Seats: 180 Delete

Modal card



INDIGO ₹10,000.00
goa chennai
Departure 20 Dec 2025, 06:30 PM
Arrival 20 Dec 2025
Seats: 147

INDIGO ₹10,000.00
Arrival 26 Dec 2025, 08:45 PM
Delete

Confirm Deletion

Do you really want to delete this flight?

This action cannot be undone.

Cancel Delete

INDIGO
goa chennai
Departure 25 Dec 2025, 06:30 PM
Arrival 26 Dec 2025, 08:45 PM
Seats: 180 Delete

₹10,000.00
goa chennai
Departure 25 Dec 2025, 06:30 PM
Arrival 26 Dec 2025, 08:45 PM
Seats: 180 Delete

Optimized Docker file:

Example for flight service:

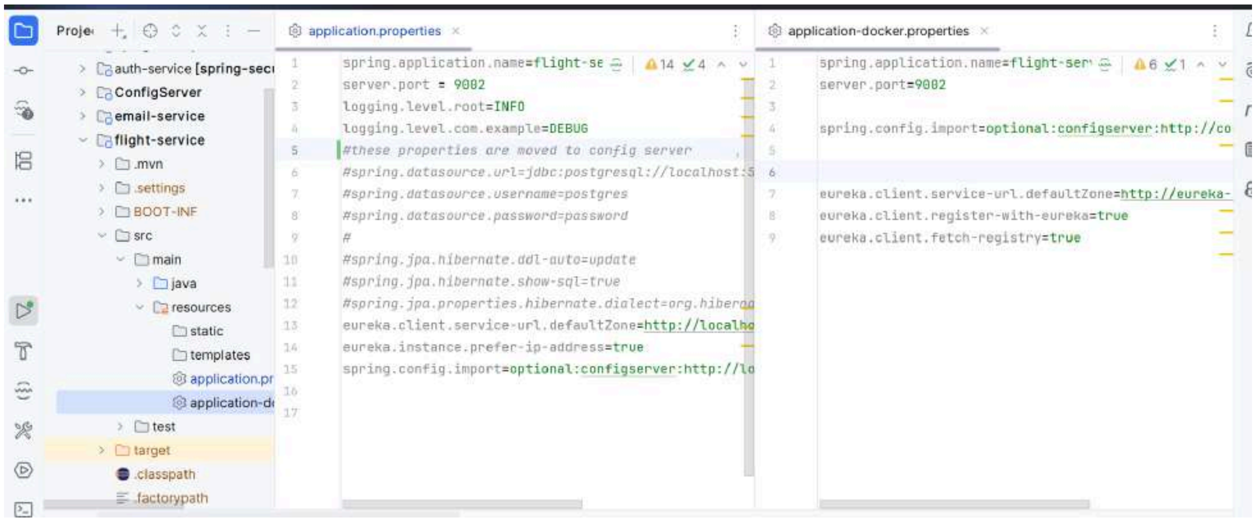
```
FROM eclipse-temurin:17-jdk

WORKDIR /app

COPY target/*.jar flight-service.jar

EXPOSE 9002

ENTRYPOINT ["java", "-jar", "flight-service.jar"]
```

Star-service.cmd file:

```
java -jar service-registry\target\service-registry-0.0.1-SNAPSHOT.jar
java -jar ConfigServer\target\ConfigServer-0.0.1-SNAPSHOT.jar
java -jar api-gateway\target\api-gateway-0.0.1-SNAPSHOT.jar
java -jar auth-service\target\spring-security-own-0.0.1-SNAPSHOT.jar
java -jar flight-service\target\flight-service-0.0.1-SNAPSHOT.jar
java -jar passenger-service\target\passenger-service-0.0.1-SNAPSHOT.jar
java -jar ticket-service\target\ticket-service-0.0.1-SNAPSHOT.jar
java -jar email-service\target\email-service-0.0.1-SNAPSHOT.jar
```

2 Property files:

1. For Docker
2. Local



```
spring.application.name=flight-se
server.port = 9002
logging.level.root=INFO
logging.level.com.example=DEBUG
#these properties are moved to config server
#spring.datasource.url=jdbc:postgresql://localhost:5
#spring.datasource.username=postgres
#spring.datasource.password=password
#
#spring.jpa.hibernate.ddl-auto=update
#spring.jpa.hibernate.show-sql=true
#spring.jpa.properties.hibernate.dialect=org.hiberna
eureka.client.service-url.defaultZone=http://localho
eureka.instance.prefer-ip-address=true
spring.config.import=optional:configserver:http://lo
```

```
spring.application.name=flight-ser
server.port=9002

spring.config.import=optional:configserver:http://co


eureka.client.service-url.defaultZone=http://eureka-
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
```

`application-docker.properties file for docker purpose`

In docker-compose.yml file this is how we use the docker profile:

```
flight-service:
 build: ./flight-service
 container_name: flight-service
 ports:
   - "9002:9002"
 environment:
   SPRING_PROFILES_ACTIVE: docker
 depends_on:
   eureka-server:
     condition: service_healthy
   config-server:
     condition: service_healthy
   postgres-flight:
     condition: service_started
 networks:
   - app-net
```

SWOT Analysis – Flight Ticket Booking Application

Strengths

- Microservices-based architecture (Eureka, Config Server, API Gateway)
- Secure authentication using **Spring Security + JWT**
- Clear separation of services (Auth, Flight, Passenger, Ticket, Email)
- Admin features for adding and managing flights

Weaknesses

- Limited frontend features and UI polish
- No real payment gateway integration
- Basic error handling and validation
- High dependency on multiple services running together

Opportunities

- Integration of payment gateways (Razorpay/Stripe)
- Seat locking and dynamic pricing
- Real-time notifications using WebSockets
- Docker + Kubernetes deployment
- Advanced search, filters, and recommendations

Threats

- System failure if a critical microservice goes down
- Security risks if JWT handling is misconfigured

- Performance issues under high traffic
- Data inconsistency in distributed services