Flight Reservation System (monolithic Spring Boot app) Case Study

project structure

```
com.example.flightreservation
|
├── entity
|   ├── Flight.java
|   └── Reservation.java
|
├── repository
|   ├── FlightRepository.java
|   └── ReservationRepository.java
|
├── service
|   ├── FlightService.java
|   └── ReservationService.java
|
├── controller
|   ├── FlightController.java
|   └── ReservationController.java
|
├── exception
|   ├── FlightNotFoundException.java
|   ├── NotEnoughSeatsException.java
|   └── GlobalExceptionHandler.java
```

```
│
├── config
│   └── SwaggerConfig.java
│
├── FlightReservationApplication.java
└── application.properties
```

Flight.java (Entity)

```
package com.example.flightreservation.entity;
import jakarta.persistence.*;
import java.time.LocalDateTime;
import java.util.List;
@Entity
public class Flight {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true, nullable = false)
    private String flightNumber;

    private String origin;
    private String destination;
    private LocalDateTime departureTime;
    private int seatsAvailable;
    @OneToMany(mappedBy = "flight", cascade = CascadeType.ALL)
    private List<Reservation> reservations;

    // Getters and Setters
}
```

Reservation.java (Entity)

```
package com.example.flightreservation.entity;
import jakarta.persistence.*;
import java.time.LocalDateTime;

@Entity
public class Reservation {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String passengerName;
    private String passengerEmail;
    private int seatsBooked;
    private LocalDateTime reservedAt;

    @ManyToOne
    @JoinColumn(name = "flight_id")
    private Flight flight;
}
```

Repositories

FlightRepository.java

```
package com.example.flightreservation.repository;
import com.example.flightreservation.entity.Flight;
import org.springframework.data.jpa.repository.JpaRepository;
public interface FlightRepository extends JpaRepository<Flight, Long> {
    boolean existsByFlightNumber(String flightNumber);
}
```

ReservationRepository.java

```
package com.example.flightreservation.repository;
import com.example.flightreservation.entity.Reservation;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;
public interface ReservationRepository extends JpaRepository<Reservation, Long> {
    List<Reservation> findByFlightId(Long flightId);
}
```

Exceptions

FlightNotFoundException.java

```
package com.example.flightreservation.exception;
public class FlightNotFoundException extends RuntimeException {
    public FlightNotFoundException(String message) {
        super(message);
    }
}
```

```
}
```

NotEnoughSeatsException.java

java

CopyEdit

```
package com.example.flightreservation.exception;


public class NotEnoughSeatsException extends RuntimeException {
    public NotEnoughSeatsException(String message) {
        super(message);
    }
}
```

GlobalExceptionHandler.java

```
package com.example.flightreservation.exception;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
@RestControllerAdvice
public class GlobalExceptionHandler {


    @ExceptionHandler(FlightNotFoundException.class)
    public ResponseEntity<String> handleFlightNotFound(FlightNotFoundException ex) {
        return ResponseEntity.status(404).body(ex.getMessage());
    }


    @ExceptionHandler(NotEnoughSeatsException.class)
    public ResponseEntity<String> handleNotEnoughSeats(NotEnoughSeatsException
ex) {
        return ResponseEntity.badRequest().body(ex.getMessage());
    }
}
```

Services

FlightService.java

```
package com.example.flightreservation.service;
import com.example.flightreservation.entity.Flight;
import com.example.flightreservation.exception.FlightNotFoundException;
import com.example.flightreservation.repository.FlightRepository;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class FlightService {

    private final FlightRepository flightRepo;

    public FlightService(FlightRepository flightRepo) {
        this.flightRepo = flightRepo;
    }

    public Flight addFlight(Flight flight) {
        return flightRepo.save(flight);
    }

    public List<Flight> getAllFlights() {
        return flightRepo.findAll();
    }
    public Flight getFlightById(Long id) {
        return flightRepo.findById(id).orElseThrow(() -> new
FlightNotFoundException("Flight not found: " + id));
    }

    public Flight updateFlight(Long id, Flight updated) {
        Flight existing = getFlightById(id);
        existing.setFlightNumber(updated.getFlightNumber());
        existing.setOrigin(updated.getOrigin());
        existing.setDestination(updated.getDestination());
        existing.setDepartureTime(updated.getDepartureTime());
        existing.setSeatsAvailable(updated.getSeatsAvailable());
        return flightRepo.save(existing);
    }

    public void deleteFlight(Long id) {
        flightRepo.deleteById(id);
    }
}
```

ReservationService.java

```
package com.example.flightreservation.service;
import com.example.flightreservation.entity.Flight;
import com.example.flightreservation.entity.Reservation;
import com.example.flightreservation.exception.FlightNotFoundException;
import com.example.flightreservation.exception.NotEnoughSeatsException;
import com.example.flightreservation.repository.FlightRepository;
import com.example.flightreservation.repository.ReservationRepository;
import org.springframework.stereotype.Service;

import java.time.LocalDateTime;
import java.util.List;

@Service
public class ReservationService {

    private final ReservationRepository reservationRepo;
    private final FlightRepository flightRepo;

    public ReservationService(ReservationRepository reservationRepo, FlightRepository
flightRepo) {
        this.reservationRepo = reservationRepo;
        this.flightRepo = flightRepo;
    }

    public Reservation reserveSeat(Long flightId, Reservation reservation) {
        Flight flight = flightRepo.findById(flightId)
                .orElseThrow(() -> new FlightNotFoundException("Flight not found"));

        if (reservation.getSeatsBooked() > flight.getSeatsAvailable()) {
            throw new NotEnoughSeatsException("Not enough seats available");
        }

        flight.setSeatsAvailable(flight.getSeatsAvailable() - reservation.getSeatsBooked());
        reservation.setFlight(flight);
        reservation.setReservedAt(LocalDateTime.now());

        flightRepo.save(flight);
        return reservationRepo.save(reservation);
    }

    public List<Reservation> getAllReservations() {
        return reservationRepo.findAll();
    }

    public List<Reservation> getReservationsForFlight(Long flightId) {
        return reservationRepo.findByFlightId(flightId);
    }

    public void cancelReservation(Long reservationId) {
        Reservation reservation = reservationRepo.findById(reservationId)
                .orElseThrow(() -> new RuntimeException("Reservation not found"));

        Flight flight = reservation.getFlight();
        flight.setSeatsAvailable(flight.getSeatsAvailable() + reservation.getSeatsBooked());
        flightRepo.save(flight);

        reservationRepo.deleteById(reservationId);
    }
}
```

Controllers

FlightController.java

```java
package com.example.flightreservation.controller;
import com.example.flightreservation.entity.Flight;
import com.example.flightreservation.service.FlightService;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@RequestMapping("/api/flights")
public class FlightController {

    private final FlightService service;

    public FlightController(FlightService service) {
        this.service = service;
    }

    @PostMapping
    public ResponseEntity<Flight> addFlight(@RequestBody Flight flight) {
        return ResponseEntity.ok(service.addFlight(flight));
    }

    @GetMapping
    public ResponseEntity<List<Flight>> getAllFlights() {
        return ResponseEntity.ok(service.getAllFlights());
    }
    @GetMapping("/{id}")
    public ResponseEntity<Flight> getFlightById(@PathVariable Long id) {
        return ResponseEntity.ok(service.getFlightById(id));
    }

    @PutMapping("/{id}")
    public ResponseEntity<Flight> updateFlight(@PathVariable Long id, @RequestBody
Flight updated) {
        return ResponseEntity.ok(service.updateFlight(id, updated));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteFlight(@PathVariable Long id) {
        service.deleteFlight(id);
        return ResponseEntity.noContent().build();
    }
}
```

ReservationController.java

```
package com.example.flightreservation.controller;
import com.example.flightreservation.entity.Reservation;
import com.example.flightreservation.service.ReservationService;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@RequestMapping("/api/reservations")
public class ReservationController {
    private final ReservationService service;

    public ReservationController(ReservationService service) {
        this.service = service;
    }

    @PostMapping("/{flightId}")
    public ResponseEntity<Reservation> makeReservation(@PathVariable Long flightId,
@RequestBody Reservation reservation) {
        return ResponseEntity.ok(service.reserveSeat(flightId, reservation));
    }

    @GetMapping
    public ResponseEntity<List<Reservation>> getAllReservations() {
        return ResponseEntity.ok(service.getAllReservations());
    }

    @GetMapping("/flight/{flightId}")
    public ResponseEntity<List<Reservation>> getReservationsByFlight(@PathVariable
Long flightId) {
        return ResponseEntity.ok(service.getReservationsForFlight(flightId));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> cancelReservation(@PathVariable Long id) {
        service.cancelReservation(id);
        return ResponseEntity.noContent().build();
    }
}
```

---

application.properties

```
spring.datasource.url=jdbc:h2:mem:flightdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
spring.jpa.hibernate.ddl-auto=update
springdoc.api-docs.path=/api-docs
springdoc.swagger-ui.path=/swagger-ui.html
```

Main Class

FlightReservationApplication.java

```
package com.example.flightreservation;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class FlightReservationApplication {
    public static void main(String[] args) {
        SpringApplication.run(FlightReservationApplication.class, args);
    }
}
```
