Case Study: Flight Reservation System (Monolithic Application)

1. Project Overview You are tasked with developing a Flight Reservation System for a small airline. The system should allow:

- Flight Management
  - Add new flights
  - View all available flights
  - View details of a specific flight
  - Update flight details (origin, destination, time, seats available)
  - Delete a flight
- Reservation Management
  - Make a reservation for a specific flight
  - View all reservations
  - View reservations for a specific flight
  - Cancel a reservation (and restore seats to the flight) This is a monolithic Spring Boot application — all functionality will be in a single codebase.

2. Technology Stack

- Spring Boot (Web + Data JPA)
- H2 Database (in-memory for development)
- Springdoc OpenAPI / Swagger (API documentation)
- Maven (dependency management)
- Java 17+
- JUnit & Mockito (optional, for unit testing)

3. Entities The system will have two main entities:

1. Flight

- id — Unique identifier (auto-generated)
- flightNumber — Unique code for the flight (e.g., AI101)
- origin — Departure city/airport

• destination — Arrival city/airport

• departureTime — Date & time of departure

• seatsAvailable — Number of available seats

2. Reservation

• id — Unique identifier (auto-generated)

• passengerName — Name of the passenger

• passengerEmail — Contact email of the passenger

• seatsBooked — Number of seats booked

• reservedAt — Date & time when reservation was made

• flight — Reference to the Flight entity (Many reservations → One flight)

4. Relationships

• One Flight can have many Reservations This means:

◦ In the database, Reservation will have a flight_id foreign key.

◦ In JPA, Reservation will use @ManyToOne to Flight.

5. API Requirements Learners should create REST APIs with the following endpoints: Flight API

• POST /api/flights → Add a new flight

• GET /api/flights → Get all flights

• GET /api/flights/{id} → Get flight by ID

• PUT /api/flights/{id} → Update a flight

• DELETE /api/flights/{id} → Delete a flight Reservation API

• POST /api/reservations → Make a reservation

◦ Reduce the available seats in the flight

◦ Reject reservation if seats are not enough

• GET /api/reservations → Get all reservations

• GET /api/reservations/flight/{flightId} → Get reservations for a specific flight

- DELETE /api/reservations/{id} → Cancel a reservation
  - Add back seats to the flight

6. Business Rules

- When making a reservation:
  - Check if the flight exists.
  - Ensure seats requested ≤ seats available.
  - Reduce seat count if successful.
- When canceling a reservation:
  - Add the booked seats back to the flight.
- A flight cannot have a negative number of seats.
- Flight numbers should be unique.

Entity class:

Flight.java:

```java
package com.example.flightreservation.entity;

import jakarta.persistence.*;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

@Entity
public class Flight {

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

    @Column(unique = true, nullable = false)
```

```java
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 = new ArrayList<>();


// Constructors

public Flight() {}


// Getters and Setters

public Long getId() {

   return id;

}


public String getFlightNumber() {

   return flightNumber;

}


public void setFlightNumber(String flightNumber) {

   this.flightNumber = flightNumber;

}


public String getOrigin() {

   return origin;

}
```

```java
public void setOrigin(String origin) {

    this.origin = origin;

}


public String getDestination() {

    return destination;

}


public void setDestination(String destination) {

    this.destination = destination;

}


public LocalDateTime getDepartureTime() {

    return departureTime;

}


public void setDepartureTime(LocalDateTime departureTime) {

    this.departureTime = departureTime;

}


public int getSeatsAvailable() {

    return seatsAvailable;

}


public void setSeatsAvailable(int seatsAvailable) {

    this.seatsAvailable = seatsAvailable;

}


public List<Reservation> getReservations() {

    return reservations;
    }

    public void setReservations(List<Reservation> reservations) {
        this.reservations = reservations;
    }
}
```

Reservation.java:

```
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", nullable = false)
    private Flight flight;

    // Constructors
```

```java
public Reservation() {}


// Getters and Setters

public Long getId() {

    return id;

}


public String getPassengerName() {

    return passengerName;

}


public void setPassengerName(String passengerName) {

    this.passengerName = passengerName;

}


public String getPassengerEmail() {

    return passengerEmail;

}


public void setPassengerEmail(String passengerEmail) {

    this.passengerEmail = passengerEmail;

}


public int getSeatsBooked() {

    return seatsBooked;

}


public void setSeatsBooked(int seatsBooked) {

    this.seatsBooked = seatsBooked;

}
```

```java
    public LocalDateTime getReservedAt() {
        return reservedAt;
    }

    public void setReservedAt(LocalDateTime reservedAt) {
        this.reservedAt = reservedAt;
    }

    public Flight getFlight() {
        return flight;
    }

    public void setFlight(Flight flight) {
        this.flight = flight;
    }
}
```

Repository:

FlightRepository:

```java
package com.example.flightreservation.repository;

import com.example.flightreservation.entity.Flight;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface FlightRepository extends JpaRepository<Flight, Long> {

    Optional<Flight> findByFlightNumber(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);

}
```

Service:

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.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;


import java.util.List;


@Service

public class FlightService {


    @Autowired
    private FlightRepository flightRepository;

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

    public List<Flight> getAllFlights() {
        return flightRepository.findAll();
    }

    public Flight getFlightById(Long id) {
        return flightRepository.findById(id)
                .orElseThrow(() -> new FlightNotFoundException("Flight not found with ID: " + id));
    }

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

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

ReservationService:

```
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.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;


import java.time.LocalDateTime;

import java.util.List;


@Service

public class ReservationService {


   @Autowired

   private ReservationRepository reservationRepository;


   @Autowired

   private FlightRepository flightRepository;


   public Reservation makeReservation(Long flightId, Reservation reservation) {

       Flight flight = flightRepository.findById(flightId)

               .orElseThrow(() -> new FlightNotFoundException("Flight not found with ID: " + flightId));


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

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

    // Save both
    flightRepository.save(flight);
    return reservationRepository.save(reservation);
}

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

public List<Reservation> getReservationsByFlightId(Long flightId) {
    return reservationRepository.findByFlightId(flightId);
}

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

    Flight flight = reservation.getFlight();

    // Restore seats
    flight.setSeatsAvailable(flight.getSeatsAvailable() + reservation.getSeatsBooked());

    // Save flight & delete reservation
```

```
        flightRepository.save(flight);

        reservationRepository.deleteById(reservationId);

    }

}
```

Controller:

FlightController:

```
package com.example.flightreservation.controller;


import com.example.flightreservation.entity.Flight;

import com.example.flightreservation.repository.FlightRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.*;


import java.util.List;

import java.util.Optional;


@RestController

@RequestMapping("/api/flights")

public class FlightController {


    @Autowired

    private FlightRepository flightRepository;


    @PostMapping

    public Flight addFlight(@RequestBody Flight flight) {

        return flightRepository.save(flight);

    }


    @GetMapping

    public List<Flight> getAllFlights() {
        return flightRepository.findAll();

    }


    @GetMapping("/{id}")

    public Optional<Flight> getFlightById(@PathVariable Long id) {

        return flightRepository.findById(id);

    }


    @PutMapping("/{id}")

    public Flight updateFlight(@PathVariable Long id, @RequestBody Flight updatedFlight) {

        return flightRepository.findById(id).map(flight -> {

            flight.setFlightNumber(updatedFlight.getFlightNumber());

            flight.setOrigin(updatedFlight.getOrigin());

            flight.setDestination(updatedFlight.getDestination());

            flight.setDepartureTime(updatedFlight.getDepartureTime());

            flight.setSeatsAvailable(updatedFlight.getSeatsAvailable());

            return flightRepository.save(flight);

        }).orElseThrow(() -> new RuntimeException("Flight not found with ID: " + id));

    }


    @DeleteMapping("/{id}")

    public void deleteFlight(@PathVariable Long id) {

        flightRepository.deleteById(id);

    }

}
```

ReservationController:

```
package com.example.flightreservation.controller;
```

```java
import com.example.flightreservation.entity.Reservation;

import com.example.flightreservation.service.ReservationService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.*;


import java.util.List;


@RestController

@RequestMapping("/api/reservations")

public class ReservationController {


  @Autowired

  private ReservationService reservationService;


  @PostMapping("/flight/{flightId}")

  public Reservation makeReservation(@PathVariable Long flightId, @RequestBody Reservation reservation) {

    return reservationService.makeReservation(flightId, reservation);

  }


  @GetMapping

  public List<Reservation> getAllReservations() {

    return reservationService.getAllReservations();

  }


  @GetMapping("/flight/{flightId}")

  public List<Reservation> getReservationsByFlightId(@PathVariable Long flightId) {

    return reservationService.getReservationsByFlightId(flightId);

  }


  @DeleteMapping("/{reservationId}")
```

```java
    public void cancelReservation(@PathVariable Long reservationId) {

        reservationService.cancelReservation(reservationId);

    }

}
```