Case Study: Flight Reservation System (Monolithic Application)

Develop a monolithic Spring Boot application for a Flight Reservation System that enables both flight and reservation management. The system should allow users to add, view, update, and delete flights, as well as make, view, and cancel reservations while managing seat availability.

//application.properties

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

// Flight.java

```
package com.example.entity;
import jakarta.persistence.*;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;
```

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Flight {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(unique = true)
    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<>();
}
```

// Reservation.java

```
package com.example.entity;
import jakarta.persistence.*;
import java.time.LocalDateTime;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

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;

}
```

// FlightRepository.java

```
package com.example.repository;

import com.example.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.repository;
import com.example.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);
}
```

// FlightNotFoundException.java

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

// NotEnoughSeatsException.java

```
package com.example.exception;
public class NotEnoughSeatsException extends RuntimeException {
    public NotEnoughSeatsException(String msg) {
        super(msg);
    }
}

// FlightService.java
package com.example.service;
import com.example.entity.Flight;
import com.example.exception.FlightNotFoundException;
import com.example.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 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 with id: " + id));
```

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

//  ReservationService.java
package com.example.service;
import com.example.entity.Flight;
import com.example.entity.Reservation;
import com.example.exception.FlightNotFoundException;
import com.example.exception.NotEnoughSeatsException;
import com.example.repository.FlightRepository;
import com.example.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 FlightRepository flightRepo;
    @Autowired
    private ReservationRepository reservationRepo;
    public Reservation makeReservation(Long flightId, Reservation reservation) {
        Flight flight = flightRepo.findById(flightId)
                .orElseThrow(() -> new FlightNotFoundException("Flight not found"));
        if (flight.getSeatsAvailable() < reservation.getSeatsBooked()) {
            throw new NotEnoughSeatsException("Not enough seats available");
        }
        flight.setSeatsAvailable(flight.getSeatsAvailable() - reservation.getSeatsBooked());
        reservation.setReservedAt(LocalDateTime.now());
        reservation.setFlight(flight);
        flightRepo.save(flight);
        return reservationRepo.save(reservation);
    }
    public List<Reservation> getAllReservations() {
        return reservationRepo.findAll();
    }
    public List<Reservation> getReservationsByFlightId(Long flightId) {
        return reservationRepo.findByFlightId(flightId);
    }
    public void cancelReservation(Long id) {
        Reservation res = reservationRepo.findById(id)
                .orElseThrow(() -> new RuntimeException("Reservation not found"));
        Flight flight = res.getFlight();
        flight.setSeatsAvailable(flight.getSeatsAvailable() + res.getSeatsBooked());
        flightRepo.save(flight);
        reservationRepo.deleteById(id);
    }
}
```

// FlightController.java

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

    @Autowired
    private FlightService flightService;
    @PostMapping
    public ResponseEntity<Flight> addFlight(@RequestBody Flight flight) {
        return ResponseEntity.ok(flightService.addFlight(flight));
    }
    @GetMapping
    public ResponseEntity<List<Flight>> getAllFlights() {
        return ResponseEntity.ok(flightService.getAllFlights());
    }
    @GetMapping("/{id}")
    public ResponseEntity<Flight> getFlightById(@PathVariable Long id) {
        return ResponseEntity.ok(flightService.getFlightById(id));
    }
    @PutMapping("/{id}")
    public ResponseEntity<Flight> updateFlight(@PathVariable Long id, @RequestBody Flight flight) {
        return ResponseEntity.ok(flightService.updateFlight(id, flight));
    }
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteFlight(@PathVariable Long id) {
        flightService.deleteFlight(id);
        return ResponseEntity.ok().build();
    }
}
```

```java
// ReservationController.java
package com.example.controller;
import com.example.entity.Reservation;
import com.example.service.ReservationService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
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 ResponseEntity<Reservation> makeReservation(
            @PathVariable Long flightId,
            @RequestBody Reservation reservation) {
        return ResponseEntity.ok(reservationService.makeReservation(flightId, reservation));
    }
    @GetMapping
    public ResponseEntity<List<Reservation>> getAllReservations() {
        return ResponseEntity.ok(reservationService.getAllReservations());
    }
    @GetMapping("/flight/{flightId}")
    public ResponseEntity<List<Reservation>> getReservationsByFlight(@PathVariable Long flightId) {

        return ResponseEntity.ok(reservationService.getReservationsByFlightId(flightId));

    }

    @DeleteMapping("/{id}")

    public ResponseEntity<Void> cancelReservation(@PathVariable Long id) {

        reservationService.cancelReservation(id);

        return ResponseEntity.ok().build();

    }

}
```

Microservices

Design and develop a microservice-based Restaurant Service that independently manages its own database and API. The service should enable adding, viewing, updating, and deleting restaurant details, as well as managing menu items for each restaurant, including listing all menu items associated with a specific restaurant.

```
// application.properties

server.port=8081

spring.datasource.url=jdbc:h2:mem:restaurantdb

spring.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

springdoc.swagger-ui.path=/swagger-ui.html
```

// Restaurant.java

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


import jakarta.persistence.*;
import java.util.ArrayList;
import java.util.List;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;


@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Restaurant {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;


   private String name;
   private String location;
   private String contactNumber;


   @OneToMany(mappedBy = "restaurant", cascade = CascadeType.ALL)
   private List<MenuItem> menuItems = new ArrayList<>();
```

```
}

// MenuItem.java
package com.example.restaurant.entity;

import jakarta.persistence.*;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class MenuItem {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String description;
    private double price;

    @ManyToOne
    @JoinColumn(name = "restaurant_id")
    private Restaurant restaurant;
}


// RestaurantRepository.java
package com.example.restaurant.repository;


import com.example.restaurant.entity.Restaurant;
import org.springframework.data.jpa.repository.JpaRepository;


public interface RestaurantRepository extends JpaRepository<Restaurant, Long> {}


// MenuItemRepository.java
package com.example.restaurant.repository;


import com.example.restaurant.entity.MenuItem;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;


public interface MenuItemRepository extends JpaRepository<MenuItem, Long> {
    List<MenuItem> findByRestaurantId(Long restaurantId);
}


// RestaurantService.java
package com.example.restaurant.service;
```

```
import com.example.restaurant.entity.MenuItem;
import com.example.restaurant.entity.Restaurant;
import com.example.restaurant.repository.MenuItemRepository;
import com.example.restaurant.repository.RestaurantRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class RestaurantService {

    @Autowired
    private RestaurantRepository restaurantRepo;

    @Autowired
    private MenuItemRepository menuItemRepo;

    public Restaurant addRestaurant(Restaurant r) {
        return restaurantRepo.save(r);
    }

    public List<Restaurant> getAll() {
        return restaurantRepo.findAll();
    }
    public Restaurant getById(Long id) {
        return restaurantRepo.findById(id)
                .orElseThrow(() -> new RuntimeException("Restaurant not found"));
    }

    public MenuItem addMenuItem(Long restaurantId, MenuItem item) {
        Restaurant restaurant = getById(restaurantId);
        item.setRestaurant(restaurant);
        return menuItemRepo.save(item);
    }

    public List<MenuItem> getMenuItems(Long restaurantId) {
        return menuItemRepo.findByRestaurantId(restaurantId);
    }
}
```

// RestaurantController.java

```
package com.example.restaurant.controller;

import com.example.restaurant.entity.MenuItem;
import com.example.restaurant.entity.Restaurant;
import com.example.restaurant.service.RestaurantService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/restaurants")
public class RestaurantController {

    @Autowired
    private RestaurantService service;

    @PostMapping
    public ResponseEntity<Restaurant> add(@RequestBody Restaurant r) {
        return ResponseEntity.ok(service.addRestaurant(r));
    }

    @GetMapping
    public ResponseEntity<List<Restaurant>> all() {
        return ResponseEntity.ok(service.getAll());
    }

    @GetMapping("/{id}")
    public ResponseEntity<Restaurant> getById(@PathVariable Long id) {
        return ResponseEntity.ok(service.getById(id));
    }
    @PostMapping("/{id}/menu-items")
    public ResponseEntity<MenuItem> addMenu(@PathVariable Long id, @RequestBody MenuItem item) {
        return ResponseEntity.ok(service.addMenuItem(id, item));
    }


    @GetMapping("/{id}/menu-items")
    public ResponseEntity<List<MenuItem>> getMenu(@PathVariable Long id) {
        return ResponseEntity.ok(service.getMenuItems(id));
    }
}
```

// application.properties

```
server.port=8082
spring.datasource.url=jdbc:h2:mem:orderdb
spring.h2.console.enabled=true
spring.jpa.hibernate.ddl-auto=update
springdoc.swagger-ui.path=/swagger-ui.html
```

// Order.java

```
package com.example.order.entity;


import jakarta.persistence.*;
import java.util.ArrayList;
import java.util.List;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;

@Entity
@Table(name = "orders")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Order {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   private String customerName;
   private String customerAddress;
   private double totalAmount;
   private String status;

   @OneToMany(mappedBy = "order", cascade = CascadeType.ALL)
   private List<OrderItem> items = new ArrayList<>();
}
```

```java
// OrderItem.java
package com.example.order.entity;

import jakarta.persistence.*;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class OrderItem {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private Long menuItemId;
    private int quantity;
    private double price;

    @ManyToOne
    @JoinColumn(name = "order_id")
    private Order order;
}
```

```
// OrderRepository.java
package com.example.order.repository;

import com.example.order.entity.Order;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface OrderRepository extends JpaRepository<Order, Long> {
    List<Order> findByCustomerName(String customerName);
}
```

```
// application.properties
server.port=8083
spring.datasource.url=jdbc:h2:mem:deliverydb
spring.h2.console.enabled=true
spring.jpa.hibernate.ddl-auto=update
springdoc.swagger-ui.path=/swagger-ui.html
```

```
// Delivery.java
package com.example.delivery.entity;

import jakarta.persistence.*;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Delivery {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private Long orderId;
    private String deliveryPersonName;
    private String deliveryStatus;
}
```