

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 \leq 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.

7. Suggested Implementation Steps 1. Setup Project

- Create a Spring Boot project with required dependencies.
- Configure application.properties for H2 database.

2. Create Entities

- Define Flight and Reservation with appropriate JPA annotations.
- Set up relationships using @ManyToOne.3.

Create Repositories

- Extend JpaRepository for both entities.

4. Write Services

- Business logic for managing flights and reservations.
- Handle seat availability logic in the reservation service.

5. Write Controllers

- Map endpoints to service methods.
- Use ResponseEntity for proper HTTP status codes.

6. Exception Handling

- Create custom exceptions (e.g., FlightNotFoundException, NotEnoughSeatsException).
- Use @ControllerAdvice for global exception handling.

7. Swagger Integration

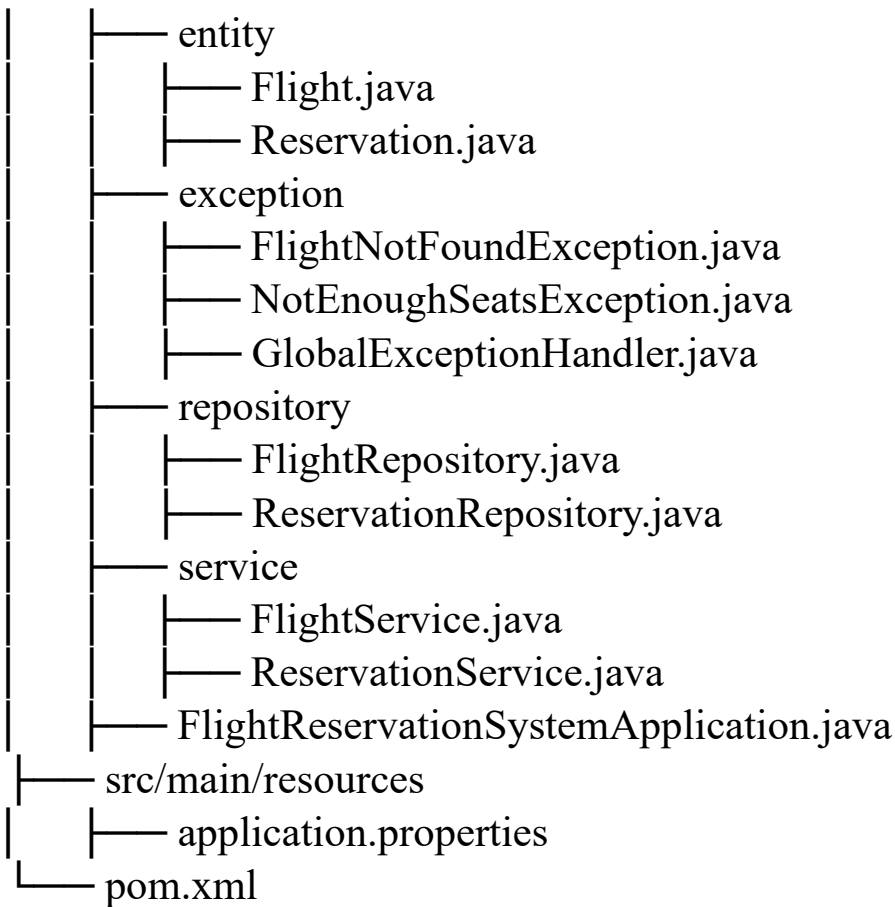
- Use Springdoc OpenAPI to generate API documentation.
- Test APIs from Swagger UI.

//Folder Structure:

```

flight-reservation-system/
├── src/main/java/com/example/flightreservation
│   ├── controller
│   │   ├── FlightController.java
│   │   └── ReservationController.java

```



//Tables creation

//flights table

```
CREATE TABLE flights ( id BIGINT PRIMARY KEY
    AUTO_INCREMENT, flight_number VARCHAR(50)
    UNIQUE NOT NULL, origin VARCHAR(100) NOT
    NULL, destination VARCHAR(100) NOT NULL,
    departure_time TIMESTAMP NOT NULL,
    seats_available INT NOT NULL
);
```

//Reservation table

```
CREATE TABLE reservations ( id BIGINT PRIMARY
    KEY AUTO_INCREMENT, passenger_name
    VARCHAR(100) NOT NULL, passenger_email
    VARCHAR(100) NOT NULL, seats_booked INT NOT
    NULL,
    reserved_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, flight_id
    BIGINT,
    CONSTRAINT fk_flight FOREIGN KEY (flight_id) REFERENCES flights(id)
);
```

//pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
  4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>flight-reservation-system</artifactId>
  <version>1.0.0</version>
  <properties>
    <java.version>17</java.version>
    <spring-boot.version>3.2.5</spring-boot.version>
  </properties>

  <dependencies>
    <!-- Spring Boot Starter Web -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- Spring Boot Starter Data JPA -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <!-- H2 Database -->
    <dependency>
      <groupId>com.h2database</groupId>
      <artifactId>h2</artifactId>
      <scope>runtime</scope>
    </dependency>

    <!-- Swagger / OpenAPI -->
    <dependency>
      <groupId>org.springdoc</groupId>
      <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
      <version>2.5.0</version>
    </dependency>

    <!-- Lombok (optional for getter/setter/constructor generation) -->
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>
  </dependencies>
</project>
```

```

</dependency>

<!-- Test Dependencies -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
</dependencies>

<build>
    <plugins>
        <!-- Spring Boot Maven Plugin -->
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
</project>

```

//application.properties

```

spring.datasource.url=jdbc:h2:mem:flightdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=root spring.datasource.password=root

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=update spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

```

//Entities //Flight.java

```

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,
orphanRemoval = true)

private List<Reservation> reservations;

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getFlightNumber() {
    return flightNumber;
}

public void setFlightNumber(String flightNumber) {
    this.flightNumber = flightNumber;
}

public String getOrigin() {
    return origin;
}

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;

public Reservation() {
}

public Reservation(String passengerName, String passengerEmail, int seatsBooked, LocalDateTime
reservedAt, Flight flight) {
    this.passengerName = passengerName;
    this.passengerEmail = passengerEmail;
    this.seatsBooked = seatsBooked;
    this.reservedAt = reservedAt;
    this.flight = flight;
}

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = 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;
}

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;
    }
}

```

//Repositories //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);
}

```

//Services

//FlightService. java

```

package com.example.flightreservation.service;

```

```

import com.example.flightreservation.entity.Flight;

import com.example.flightreservation.repository.FlightRepository;
import org.springframework.http.HttpStatus;

import org.springframework.stereotype.Service;

import org.springframework.web.server.ResponseStatusException;
import java.util.List;

@Service
public class FlightService {
    private final FlightRepository flightRepository;

    public FlightService(FlightRepository flightRepository) {
        this.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 ResponseStatusException(
                HttpStatus.NOT_FOUND, "Flight not found with ID: " + id
            ));
    }

    public Flight updateFlight(Long id, Flight updatedFlight) {
        Flight flight = getFlightById(id);
        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) {
        Flight flight = getFlightById(id);
        flightRepository.delete(flight);
    }
}

```

//ReservationService.java

```
package com.example.flightreservation.service;

import com.example.flightreservation.entity.Flight;

import com.example.flightreservation.entity.Reservation;

import com.example.flightreservation.repository.FlightRepository;

import com.example.flightreservation.repository.ReservationRepository;
import org.springframework.http.HttpStatus;

import org.springframework.stereotype.Service;

import org.springframework.web.server.ResponseStatusException;

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

@Service
public class ReservationService {
    private final ReservationRepository reservationRepository;
    private final FlightRepository flightRepository;
    public ReservationService(ReservationRepository reservationRepository, FlightRepository
flightRepository) {
        this.reservationRepository = reservationRepository;
        this.flightRepository = flightRepository;
    }

    public Reservation makeReservation(Long flightId, Reservation reservation) {
        Flight flight = flightRepository.findById(flightId)
        .orElseThrow(() -> new ResponseStatusException(
        HttpStatus.NOT_FOUND, "Flight not found with ID: " + flightId ));

        if (reservation.getSeatsBooked() > flight.getSeatsAvailable()) {
            throw new ResponseStatusException(
                HttpStatus.BAD_REQUEST, "Not enough seats available."
            )
        }

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

        reservation.setFlight(flight);

        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 ResponseStatusException(
                HttpStatus.NOT_FOUND, "Reservation not found with ID: " + reservationId
            ));

        Flight flight = reservation.getFlight();

        flight.setSeatsAvailable(flight.getSeatsAvailable() + reservation.getSeatsBooked());

        reservationRepository.delete(reservation); flightRepository.save(flight);
    }
}

```

//Controllers //FlightController.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 flightService;

    public FlightController(FlightService flightService) {
        this.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.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 reservationService;

    public ReservationController(ReservationService reservationService) {
        this.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>> getReservationsByFlightId(@PathVariable Long flightId)
{
    return ResponseEntity.ok(reservationService.getReservationsByFlightId(flightId));
}

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

```

2. Microservices Overview

We will have **three microservices**, each running independently, with its own database and API.

1. Restaurant Service

Responsibilities:

- Manage restaurant details.
- Manage menu items for each restaurant. **Core Features:**
- Add, view, update, delete restaurants.
- Add, view, update, delete menu items for a restaurant.
- List all menu items for a restaurant.

Entity Examples:• Restaurant

- id (PK)
- name

- location
- contactNumber
- **MenuItem**
- id (PK)
- restaurantId (FK to Restaurant)
- name
- description
- price

API Examples:

- POST /restaurants
- GET /restaurants
- GET /restaurants/{id}
- POST /restaurants/{id}/menu-items
- GET /restaurants/{id}/menu-items

//Tables creation

//Restaurant table

```
CREATE TABLE restaurants ( id BIGINT PRIMARY
    KEY AUTO_INCREMENT, name VARCHAR(100)
    NOT NULL, location VARCHAR(100) NOT NULL,
    contact_number VARCHAR(20)
);
```

//Menu Item table

```
CREATE TABLE menu_items ( id BIGINT PRIMARY
    KEY AUTO_INCREMENT,
    restaurant_id BIGINT NOT NULL,
    name VARCHAR(100) NOT NULL,
    description VARCHAR(255), price
    DECIMAL(10,2) NOT NULL,
    CONSTRAINT fk_restaurant FOREIGN KEY (restaurant_id) REFERENCES restaurants(id)
);
```

//Entities Restaurant.java

```
package com.example.restaurantservice.entity;
```

```
import jakarta.persistence.*;
```

```
@Entity
```



```
public class Restaurant {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name; private
    String location; private String
    contactNumber; public
    Restaurant() {}

    public Restaurant(String name, String location, String contactNumber) {
        this.name = name;
        this.location = location;
        this.contactNumber = contactNumber;
    }

    // Getters and Setters

    public Long getId() {

        return id;

    }

    public void setId(Long id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getLocation() {

        return location;

    }

    public void setLocation(String location) {
```

```
        this.location = location;
    }
    public String getContactNumber() {
        return contactNumber;
    }
    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }
}
```

//MenuItem.java

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

import jakarta.persistence.*;

@Entity
public class MenuItem {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private Long restaurantId;
    private String name; private
    String description; private
    double price; public
    MenuItem() {}

    public MenuItem(Long restaurantId, String name, String description, double price) {
        this.restaurantId = restaurantId;
        this.name = name;
        this.description = description;
        this.price = price;
    }

    // Getters & Setters

    public Long getId() {

        return id;

    }
}
```

```
public void setId(Long id) {  
    this.id = id;  
}  
  
public Long getRestaurantId() {  
    return restaurantId;  
}  
  
public void setRestaurantId(Long restaurantId) {  
    this.restaurantId = restaurantId;  
}  
  
public String getName() {  
    return name;  
}  
  
public void setName(String name) {  
    this.name = name;  
}  
  
public String getDescription() {  
    return description;  
}  
  
public void setDescription(String description) {  
    this.description = description;  
}  
  
public double getPrice() {  
    return price;  
}  
public void setPrice(double price) {  
    this.price = price;  
}  
}
```

//Repositories //RestaurantRepository.java

```
package com.example.restaurant.service.repository;
```

```
import com.example.restaurant.service.entity.Restaurant;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
```

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

//MenuItemRepository.java

```
package com.example.restaurant.service.repository;
```

```
import com.example.restaurant.service.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);  
}
```

//Service //RestaurantService.java

```
package com.example.restaurant.service.service;
```

```
import com.example.restaurant.service.entity.MenuItem;
```

```
import com.example.restaurant.service.entity.Restaurant;
```

```
import com.example.restaurant.service.repository.MenuItemRepository;
```

```
import com.example.restaurant.service.repository.RestaurantRepository;
```

```
import org.springframework.http.HttpStatus;
```

```
import org.springframework.stereotype.Service;
```

```
import org.springframework.web.server.ResponseStatusException;
```

```
import java.util.List;
```

```
@Service
```

```
public class RestaurantService {
```

```
    private final RestaurantRepository restaurantRepository;
```

```

private final MenuItemRepository menuItemRepository;

public RestaurantService(RestaurantRepository restaurantRepository, MenuItemRepository
menuItemRepository) {
    this.restaurantRepository = restaurantRepository;
    this.menuItemRepository = menuItemRepository;
}

public Restaurant addRestaurant(Restaurant restaurant) {

    return restaurantRepository.save(restaurant);

}

public List<Restaurant> getAllRestaurants() {
    return restaurantRepository.findAll();
}

public Restaurant getRestaurantById(Long id) {
    return restaurantRepository.findById(id)
        .orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT_FOUND, "Restaurant not
found"));
}

public MenuItem addMenuItem(Long restaurantId, MenuItem menuItem) {
    getRestaurantById(restaurantId); // ensure restaurant exists
    menuItem.setRestaurantId(restaurantId);
    return menuItemRepository.save(menuItem);
}

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

```

//Controller //RestaurantController.java

```

package com.example.restaurantservice.controller;

import com.example.restaurantservice.entity.MenuItem;

import com.example.restaurantservice.entity.Restaurant;

import com.example.restaurantservice.service.RestaurantService;
import org.springframework.http.ResponseEntity;

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

```

```

import java.util.List;

@RestController
@RequestMapping("/restaurants")

public class RestaurantController {

    private final RestaurantService restaurantService;

    public RestaurantController(RestaurantService restaurantService) {
        this.restaurantService = restaurantService;
    }

    @PostMapping
    public ResponseEntity<Restaurant> addRestaurant(@RequestBody Restaurant restaurant) {
        return ResponseEntity.ok(restaurantService.addRestaurant(restaurant));
    }

    @GetMapping
    public ResponseEntity<List<Restaurant>> getAllRestaurants() {
        return ResponseEntity.ok(restaurantService.getAllRestaurants());
    }

    @GetMapping("/{id}")
    public ResponseEntity<Restaurant> getRestaurantById(@PathVariable Long id) {
        return ResponseEntity.ok(restaurantService.getRestaurantById(id));
    }

    @PostMapping("/{id}/menu-items")
    public ResponseEntity<MenuItem> addMenuItem(@PathVariable Long id, @RequestBody MenuItem menuItem) {
        return ResponseEntity.ok(restaurantService.addMenuItem(id, menuItem));
    }

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

```

2. Order Service

Responsibilities:

- Handle customer orders.
- Track order status (PLACED, PREPARING, DELIVERED, CANCELED).

Features:

- Place an order for one or more menu items (fetch menu from Restaurant Service).
- View all orders for a customer.
- Update order status. **Entity Examples:**

- **Order**

- id (PK)
- customerName
- customerAddress
- totalAmount
- status

- **OrderItem**

- id (PK)
- orderId (FK to Order)
- menuItemId
- quantity
- price

API Examples:

- POST /orders (calls Restaurant Service to verify menu item availability & price)
- GET /orders/{id}
- GET /customers/{customerName}/orders
- PUT /orders/{id}/status

//Tables creation

//orders table

```
CREATE TABLE orders ( id BIGINT PRIMARY KEY
    AUTO_INCREMENT, customer_name
    VARCHAR(100) NOT NULL, customer_address
    VARCHAR(255) NOT NULL, total_amount
    DECIMAL(10,2) NOT NULL,
    status VARCHAR(20) NOT NULL
);
```

//order_items table

```
CREATE TABLE order_items ( id BIGINT PRIMARY
    KEY AUTO_INCREMENT,
    order_id BIGINT NOT NULL,
    menu_item_id BIGINT NOT NULL,
    quantity INT NOT NULL, price
    DECIMAL(10,2) NOT NULL,
```

```
    CONSTRAINT fk_order FOREIGN KEY (order_id) REFERENCES orders(id)
);
```

//Entities

//Order.java

```
package com.example.orderservice.entity;

import jakarta.persistence.*;

@Entity(name = "orders") // "order" is a reserved keyword in SQL public class
Order {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String customerName;
    private String customerAddress;

    private double totalAmount;

    @Enumerated(EnumType.STRING)
    private OrderStatus status;

    public Order() {}

    public Order(String customerName, String customerAddress, double totalAmount, OrderStatus status)
    {
        this.customerName = customerName;
        this.customerAddress = customerAddress;
        this.totalAmount = totalAmount;
        this.status = status;
    }

    // Getters & Setters

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
```



```

        this.id = id;
    }

    public String getCustomerName() {
        return customerName;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public String getCustomerAddress() {
        return customerAddress;
    }
    public void setCustomerAddress(String customerAddress) {
        this.customerAddress =customerAddress;
    }
    public double getTotalAmount() {
        return totalAmount;
    }

    public void setTotalAmount(double totalAmount) {
        this.totalAmount = totalAmount;
    }

    public OrderStatus getStatus() {
        return status;
    }
    public void setStatus(OrderStatus status) {
        this.status = status;
    }
}

```

//OrderStatus.java

```

package com.example.orderservice.entity;

public enum OrderStatus {

```

```
    PLACED,  
    PREPARING, DELIVERED,  
    CANCELED  
}
```

//OrderItem.java

```
Package com.example.orderservice.entity;
```

```
import jakarta.persistence.*;
```

```
@Entity  
public class OrderItem {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
  
    private Long id;  
  
    private Long orderId; private  
    Long menuItemId; private  
    int quantity; private double  
    price; public OrderItem() {}  
  
    public OrderItem(Long orderId, Long menuItemId, int quantity, double price) {  
        this.orderId = orderId;  
        this.menuItemId = menuItemId;  
        this.quantity = quantity;  
        this.price = price;  
    }  
  
    // Getters & Setters  
  
    public Long getId() {  
  
        return id;  
    }  
  
    public void setId(Long id) {  
  
        this.id = id;  
    }  
}
```

```
public Long getOrderId() {  
    return orderId;  
}  
  
public void setOrderId(Long orderId) {  
    this.orderId = orderId;  
}  
  
public Long getMenuItemId() {  
    return menuItemId;  
}  
  
public void setMenuItemId(Long menuItemId) {  
    this.menuItemId = menuItemId;  
}  
  
public int getQuantity() {  
    return quantity;  
}  
  
public void setQuantity(int quantity) {  
    this.quantity = quantity;  
}  
  
public double getPrice() {  
    return price;  
}  
  
public void setPrice(double price) {  
    this.price = price;  
}  
}
```

//Repositories //OrderRepository.java

```
package com.example.orderservice.repository;

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

//OrderItemRepository.java

```
package com.example.orderservice.repository;

import com.example.orderservice.entity.OrderItem;

import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface OrderItemRepository extends JpaRepository<OrderItem, Long> {
    List<OrderItem> findById(Long orderId);
}
```

//Service //OrderService.java

```
package com.example.orderservice.service;

import com.example.orderservice.entity.*;
import com.example.orderservice.repository.OrderItemRepository;
import com.example.orderservice.repository.OrderRepository;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;
import org.springframework.web.server.ResponseStatusException;

import java.util.List;

@Service
public class OrderService {

    private final OrderRepository orderRepository;

    private final OrderItemRepository orderItemRepository;
```

```

public OrderService(OrderRepository orderRepository, OrderItemRepository orderItemRepository) {
    this.orderRepository = orderRepository;
    this.orderItemRepository = orderItemRepository;
}

public Order placeOrder(Order order, List<OrderItem> items) {
    order.setStatus(OrderStatus.PLACED);

    order = orderRepository.save(order);

    double total = 0;
    for (OrderItem item : items) {
        item.setOrderId(order.getId());
        total += item.getPrice() *
            item.getQuantity();
        orderItemRepository.save(item);
    }

    order.setTotalAmount(total);
    return orderRepository.save(order);
}

public Order getOrderById(Long id) {
    return orderRepository.findById(id)
        .orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT_FOUND, "Order not found"));
}

public List<Order> getOrdersByCustomer(String customerName) {
    return orderRepository.findByCustomerName(customerName);
}

public Order updateOrderStatus(Long id, OrderStatus status) {
    Order order = getOrderById(id);
    order.setStatus(status);
    return orderRepository.save(order);
}
}

```

//Controller // OrderController.java

```

package com.example.orderservice.controller;

import com.example.orderservice.entity.*;

import com.example.orderservice.service.OrderService;
import org.springframework.http.ResponseEntity;

```

```

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

import java.util.List;

@RestController
@RequestMapping("/orders")
public class OrderController {
    private final OrderService orderService;
    public OrderController(OrderService orderService) {
        this.orderService = orderService;
    }

    @PostMapping
    public ResponseEntity<Order> placeOrder(@RequestBody OrderRequest request) {
        return ResponseEntity.ok(orderService.placeOrder(request.getOrder(), request.getItems()));
    }

    @GetMapping("/{id}")
    public ResponseEntity<Order> getOrderById(@PathVariable Long id) {
        return ResponseEntity.ok(orderService.getOrderById(id));
    }

    @GetMapping("/customers/{customerName}")
    public ResponseEntity<List<Order>> getOrdersByCustomer(@PathVariable String customerName) {
        return ResponseEntity.ok(orderService.getOrdersByCustomer(customerName));
    }

    @PutMapping("/{id}/status")
    public ResponseEntity<Order> updateOrderStatus(@PathVariable Long id, @RequestParam
    OrderStatus status) {
        return ResponseEntity.ok(orderService.updateOrderStatus(id, status));
    }
}

```

//OrderRequest.java

```

package com.example.orderservice.entity;

import java.util.List;

public class OrderRequest {

    private Order order;

    private List<OrderItem> items;

    public Order getOrder() {

```

```

        return order;
    }

    public void setOrder(Order order) {

        this.order = order;
    }

    public List<OrderItem> getItems() {
        return items;
    }
    public void setItems(List<OrderItem> items) {
        this.items = items;
    }
}

```

3. Delivery Service

Responsibilities:

- Assign delivery agents to orders.
- Track delivery status.

Core Features:

- Assign delivery person when order status becomes "PREPARING".
- Update delivery status.
- Track delivery by order ID.

Entity Examples:

- **Delivery**
 - id (PK)
 - orderId
 - deliveryPersonName
 - deliveryStatus (ASSIGNED, OUT_FOR_DELIVERY, DELIVERED)

API Examples:

- POST /deliveries (triggered when Order Service updates order to PREPARING)
- GET /deliveries/{orderId}
- PUT /deliveries/{id}/status

3. Database Design

Each microservice has its own **independent database**:

- Restaurant DB → Tables: restaurants, menu_items
- Order DB → Tables: orders, order_items
- Delivery DB → Tables: deliveries

//Tables creation

//Deliveries table

```
CREATE TABLE deliveries ( id BIGINT PRIMARY
    KEY AUTO_INCREMENT,
    order_id BIGINT NOT NULL,
    delivery_person_name VARCHAR(100),
    delivery_status VARCHAR(30) NOT NULL
);
```

//Entities //Delivery.java

```
package com.example.deliveryservice.entity;
```

```
import jakarta.persistence.*;
```

```
@Entity
```

```
public class Delivery {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
```

```
    private Long orderId;
```

```
    private String deliveryPersonName;
```

```
    private String deliveryStatus; // ASSIGNED, OUT_FOR_DELIVERY, DELIVERED
```

```
    public Delivery() {}
```

```
    public Delivery(Long orderId, String deliveryPersonName, String deliveryStatus) {
```

```
        this.orderId = orderId;
```

```
        this.deliveryPersonName = deliveryPersonName;
```

```
        this.deliveryStatus = deliveryStatus;
```

```
    }
```

```
// Getters & Setters
```

```
    public Long getId() {
```

```
        return id;
```

```
    }
```



```

public void setId(Long id) {

    this.id = id;

}

public Long getOrderId() {

    return orderId;

}

public void setOrderId(Long orderId) {

    this.orderId = orderId;

}

public String getDeliveryPersonName() {
    return deliveryPersonName;
}
public void setDeliveryPersonName(String deliveryPersonName) {
    this.deliveryPersonName =deliveryPersonName;
}

public String getDeliveryStatus() {
    return deliveryStatus;
}
public void setDeliveryStatus(String deliveryStatus) {
    this.deliveryStatus = deliveryStatus;
}

```

//Repository

//DeliveryRepository.java

```

package com.example.deliveryservice.repository;

import com.example.deliveryservice.entity.Delivery;

import org.springframework.data.jpa.repository.JpaRepository;
import java.util.Optional;

public interface DeliveryRepository extends JpaRepository<Delivery, Long> {
    Optional<Delivery> findById(Long orderId);
}

```

//Service

//DeliveryService.java

```
package com.example.deliveryservice.service;

import com.example.deliveryservice.entity.Delivery;
import com.example.deliveryservice.repository.DeliveryRepository;
import org.springframework.http.HttpStatus;

import org.springframework.stereotype.Service;

import org.springframework.web.server.ResponseStatusException;

@Service
public class DeliveryService {

    private final DeliveryRepository deliveryRepository;

    public DeliveryService(DeliveryRepository deliveryRepository) {
        this.deliveryRepository = deliveryRepository;
    }

    public Delivery assignDelivery(Delivery delivery) {
        delivery.setDeliveryStatus("ASSIGNED");
        return deliveryRepository.save(delivery);
    }

    public Delivery getDeliveryById(Long orderId) {
        return deliveryRepository.findById(orderId)
            .orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT_FOUND, "Delivery not found for order"));
    }

    public Delivery updateDeliveryStatus(Long id, String status) {
        Delivery delivery = deliveryRepository.findById(id)
            .orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT_FOUND, "Delivery not found"));
        delivery.setDeliveryStatus(status);
        return deliveryRepository.save(delivery);
    }
}
```

//Controller //DeliveryController.java

```
package com.example.deliveryservice.controller;
```

```
import com.example.deliveryservice.entity.Delivery;

import com.example.deliveryservice.service.DeliveryService;
import org.springframework.http.ResponseEntity;

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

@RestController
@RequestMapping("/deliveries")

public class DeliveryController {

    private final DeliveryService deliveryService;

    public DeliveryController(DeliveryService deliveryService) {
        this.deliveryService = deliveryService;
    }

    @PostMapping
    public ResponseEntity<Delivery> assignDelivery(@RequestBody Delivery delivery) {
        return ResponseEntity.ok(deliveryService.assignDelivery(delivery));
    }

    @GetMapping("/{orderId}")
    public ResponseEntity<Delivery> getDeliveryById(@PathVariable Long orderId) {
        return ResponseEntity.ok(deliveryService.getDeliveryById(orderId));
    }

    @PutMapping("/{id}/status")
    public ResponseEntity<Delivery> updateDeliveryStatus(@PathVariable Long id, @RequestParam
String status) {
        return ResponseEntity.ok(deliveryService.updateDeliveryStatus(id, status));
    }
}
```