Case Study: Flight Reservation System(MonolithicApplication)

1. Project Overview You are tasked with developing a Flight Reservation Systemfor 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 functionalitywill beinasingle 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

      :

Program:

Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
    <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>3.5.4</version>
            <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.wipro</groupId>
    <artifactId>Assignment1_Day14_FlightReservationSystem_Monoli
thic</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Assignment1_Day14_FlightReservationSystem_Monolithic
</name>
    <description>Flight Reservation System (Monolithic
Application)</description>
    <url/>
    <licenses>
            <license/>
    </licenses>
    <developers>
            <developer/>
    </developers>
    <scm>
            <connection/>
            <developerConnection/>
            <tag/>
            <url/>
    </scm>
```

```
<properties>
        <java.version>21</java.version>
</properties>
<dependencies>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-devtools</artifactId>
                <scope>runtime</scope>
                <optional>true</optional>
        </dependency>
        <dependency>
                <groupId>com.mysql</groupId>
                <artifactId>mysql-connector-j</artifactId>
                <scope>runtime</scope>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
        </dependency>
        <dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
     <scope>test</scope>
  </dependency>
        <dependency>
```

```
                <groupId>com.h2database</groupId>
                <artifactId>h2</artifactId>
                <scope>runtime</scope>
            </dependency>
            <dependency>
        <groupId>org.springdoc</groupId>
        <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
        <version>2.1.0</version>
       </dependency>
    </dependencies>

    <build>
            <plugins>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-
plugin</artifactId>
                </plugin>
            </plugins>
    </build>

</project>
```

Application.properties:

```
spring.application.name=Assignment1_Day14_FlightReservationSyste
m_Monolithic

spring.datasource.url=jdbc:h2:mem:flight_db
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

springdoc.api.docs.path=/api-docs
springdoc.swagger-ui.path=/swagger.ui.html

Controller
FlightController:

```
package com.example.flight.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.example.flight.entity.Flight;
import com.example.flight.service.FlightService;

@RestController
@RequestMapping("/api/flights")
public class FlightController {

    @Autowired
    private FlightService flightService;

    @PostMapping
  public ResponseEntity<Flight> createFlight(@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> getFlight(@PathVariable Long id) {
        return ResponseEntity.ok(flightService.getFlightById(id));
    }

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

ReservationController:

```
package com.example.flight.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.example.flight.entity.Reservation;
import com.example.flight.service.ReservationService;

@RestController
@RequestMapping("/api/reservations")
public class ReservationController {

    @Autowired
    private  ReservationService reservationService;

    @PostMapping
    public ResponseEntity<Reservation> createReservation(
                    @RequestParam Long flightId,
                    @RequestParam String passengerName,
                    @RequestParam String passengerEmail,
                    @RequestParam int seats) {
```

```
            return ResponseEntity.ok(
                        reservationService.makeReservation(flightId,
passengerName, passengerEmail, seats)
                        );
    }


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

}
```

Entity:

Flight.java:

```
package com.example.flight.entity;

import java.time.LocalDateTime;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="`flight`")
public class Flight {

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

```
    private Long id;

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

    @Column(nullable = false)
    private String origin;

    @Column(nullable = false)
    private String destination;

    @Column(nullable = false)
    private LocalDateTime departureTime;  // database column:
"departure_time"

    @Column(nullable = false)
    private int seatsAvailable;

    public Flight() {
            // TODO Auto-generated constructor stub
     }

     public Flight(Long id, String flightNumber, String origin, String
destination, LocalDateTime departureTime,
                    int seatsAvailable) {
            this.id = id;
            this.flightNumber = flightNumber;
            this.origin = origin;
            this.destination = destination;
            this.departureTime = departureTime;
            this.seatsAvailable = seatsAvailable;
     }

     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;
    }


}
```

Reservation.java:

```
package com.example.flight.entity;

import java.time.LocalDateTime;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.Table;

@Entity
@Table(name="`reservation`")
public class Reservation {

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

```
    @Column(nullable=false)
    private String passengerName;

    @Column(nullable=false)
   private String passengerEmail;

    @Column(nullable=false)
   private int seatsBooked;

    @Column(nullable=false)
   private LocalDateTime reservedAt;

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

   public Reservation() {
            // TODO Auto-generated constructor stub
    }

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


}
```

Service:

FlightService.java:

```
package com.example.flight.service;

import java.util.List;

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

import com.example.flight.entity.Flight;
import com.example.flight.repository.FlightRepository;

@Service
public class FlightService {

    @Autowired
    private FlightRepository flightRepository;

    public FlightService(FlightRepository flightRepository) {
      this.flightRepository = flightRepository;
   }
```

```
    public Flight addFlight(Flight flight) {
      if
(flightRepository.existsByFlightNumber(flight.getFlightNumber())) {
         throw new RuntimeException("Flight number already exists");
      }
      return flightRepository.save(flight);
   }



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



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



    public String deleteFlight(Long id) {
            if(!flightRepository.existsById(id)) {
                  throw new RuntimeException("Flight not found with
id: " + id);
            }
            flightRepository.deleteById(id);
            return "Flight id: "+id+ " Deleted Successfully";
    }
}
```

ReservationService:

```
package com.example.flight.service;

import java.time.LocalDateTime;

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

import com.example.flight.entity.Flight;
import com.example.flight.entity.Reservation;
import com.example.flight.repository.FlightRepository;
import com.example.flight.repository.ReservationRepository;

import jakarta.transaction.Transactional;

@Service
public class ReservationService {

    @Autowired
    private ReservationRepository reservationRepository;

    @Autowired
    private  FlightRepository flightRepository;

    public ReservationService(ReservationRepository
reservationRepository,
                    FlightRepository flightRepository) {
            this.reservationRepository = reservationRepository;
            this.flightRepository = flightRepository;
    }



    @Transactional
    public Reservation makeReservation(Long flightId, String
passengerName,
```

```
                String passengerEmail, int seats) {
        Flight flight = flightRepository.findById(flightId)
                .orElseThrow(() -> new
RuntimeException("Flight not found"));

        if (seats <= 0) {
                throw new RuntimeException("Seats must be
positive");
        }

        if (flight.getSeatsAvailable() < seats) {
                throw new RuntimeException("Not enough seats
available");
        }

        flight.setSeatsAvailable(flight.getSeatsAvailable() - seats);
        flightRepository.save(flight);

        Reservation reservation = new Reservation();
        reservation.setPassengerName(passengerName);
        reservation.setPassengerEmail(passengerEmail);
        reservation.setSeatsBooked(seats);
        reservation.setReservedAt(LocalDateTime.now());
        reservation.setFlight(flight);

        return reservationRepository.save(reservation);
    }



    @Transactional
      public String cancelReservation(Long reservationId) {
         Reservation reservation =
reservationRepository.findById(reservationId)
```

```
                .orElseThrow(() -> new RuntimeException("Reservation
not found"));

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

        reservationRepository.delete(reservation);

        return "Flight with Reservation Id: "+reservationId+"
Cancelled Successfully";
    }


}
```

Repository:

FlightRepository:

```
package com.example.flight.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.flight.entity.Flight;

@Repository
public interface FlightRepository extends JpaRepository<Flight,
Long>{

    boolean existsByFlightNumber(String flightNumber);
}
```

ReservationRepository:

```
package com.example.flight.repository;
```

```
import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.flight.entity.Flight;
import com.example.flight.entity.Reservation;

@Repository
public interface ReservationRepository extends
JpaRepository<Reservation, Long>{

    List<Reservation> findByFlight(Flight flight);
}
```