1 — pom.xml
xml
Copy
Edit

```
<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-monolith</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <properties>
    <java.version>17</java.version>
    <spring.boot.version>3.1.4</spring.boot.version>
  </properties>

  <dependencies>
    <!-- Spring Boot starters -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

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

    <!-- OpenAPI / Swagger UI -->
    <dependency>
      <groupId>org.springdoc</groupId>
      <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
      <version>2.1.0</version>
    </dependency>

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

```
  </dependencies>

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

2 — src/main/resources/application.properties

```
properties
Copy
Edit
spring.datasource.url=jdbc:h2:mem:flightsdb;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALS
E
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console

springdoc.api-docs.path=/v3/api-docs
springdoc.swagger-ui.path=/swagger-ui.html

server.port=8080
```

3 — Main application

src/main/java/com/example/flight/FlightReservationMonolithApplication.java

```
java
Copy
Edit
package com.example.flight;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class FlightReservationMonolithApplication {
    public static void main(String[] args) {
        SpringApplication.run(FlightReservationMonolithApplication.class, args);
    }
}
```

4 — Entities (with all getters/setters)

src/main/java/com/example/flight/entity/Flight.java

```
java
Copy
```

Edit

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

import jakarta.persistence.*;
import java.time.LocalDateTime;
import java.util.Objects;

@Entity
@Table(name = "flights", uniqueConstraints = {@UniqueConstraint(columnNames =
{"flightNumber"})})
public class Flight {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable=false, unique=true)
    private String flightNumber;
    private String origin;
    private String destination;
    private LocalDateTime departureTime;
    private Integer seatsAvailable;

    public Flight() {}

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

    // Getters and Setters (all)
    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 Integer getSeatsAvailable() { return seatsAvailable; }
    public void setSeatsAvailable(Integer seatsAvailable) { this.seatsAvailable =
seatsAvailable; }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Flight flight = (Flight) o;
        return Objects.equals(id, flight.id);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id);
    }
}
src/main/java/com/example/flight/entity/Reservation.java
java
Copy
Edit
package com.example.flight.entity;

import jakarta.persistence.*;
import java.time.LocalDateTime;
import java.util.Objects;

@Entity
@Table(name = "reservations")
public class Reservation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

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

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "flight_id")
    private Flight flight;

    public Reservation() {}

    public Reservation(String passengerName, String passengerEmail, Integer
seatsBooked, LocalDateTime reservedAt, Flight flight) {
```

```
        this.passengerName = passengerName;
        this.passengerEmail = passengerEmail;
        this.seatsBooked = seatsBooked;
        this.reservedAt = reservedAt;
        this.flight = flight;
    }

    // Getters and Setters (all)
    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 Integer getSeatsBooked() { return seatsBooked; }
    public void setSeatsBooked(Integer 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; }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Reservation that = (Reservation) o;
        return Objects.equals(id, that.id);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id);
    }
}
```

5 — Repositories

src/main/java/com/example/flight/repository/FlightRepository.java

java

Copy

Edit

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

```
import com.example.flight.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);
    boolean existsByFlightNumber(String flightNumber);
}
```

src/main/java/com/example/flight/repository/ReservationRepository.java

java

Copy

Edit

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

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

6 — Exceptions & Handler

src/main/java/com/example/flight/exception/FlightNotFoundException.java

java

Copy

Edit

```
package com.example.flight.exception;

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

src/main/java/com/example/flight/exception/NotEnoughSeatsException.java

java

Copy

Edit

```
package com.example.flight.exception;

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

src/main/java/com/example/flight/exception/GlobalExceptionHandler.java

java

Copy

Edit

```
package com.example.flight.exception;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.http.HttpStatus;
```

```
@ControllerAdvice
public class GlobalExceptionHandler {

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

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

    @ExceptionHandler(IllegalArgumentException.class)
    public ResponseEntity<String> handleBadRequest(IllegalArgumentException ex) {
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(ex.getMessage());
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> handleAll(Exception ex) {
        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Internal error: " +
ex.getMessage());
    }
}
7 — Services (business logic)
src/main/java/com/example/flight/service/FlightService.java
java
Copy
Edit
package com.example.flight.service;

import com.example.flight.entity.Flight;
import com.example.flight.exception.FlightNotFoundException;
import com.example.flight.repository.FlightRepository;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service
public class FlightService {
    private final FlightRepository flightRepository;
    public FlightService(FlightRepository flightRepository) {
        this.flightRepository = flightRepository;
    }
```

```
    public Flight createFlight(Flight flight) {
        if (flight.getSeatsAvailable() != null && flight.getSeatsAvailable() < 0) {
            throw new IllegalArgumentException("Seats cannot be negative");
        }
        if (flight.getFlightNumber() == null || flight.getFlightNumber().isBlank())
{
            throw new IllegalArgumentException("Flight number is required");
        }
        if (flightRepository.existsByFlightNumber(flight.getFlightNumber())) {
            throw new IllegalArgumentException("Flight number must be unique");
        }
        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: " + id));
    }

    @Transactional
    public Flight updateFlight(Long id, Flight updated) {
        Flight f = getFlightById(id);
        if (updated.getFlightNumber() != null &&
!updated.getFlightNumber().equals(f.getFlightNumber())) {
            if (flightRepository.existsByFlightNumber(updated.getFlightNumber())) {
                throw new IllegalArgumentException("Flight number must be unique");
            }
            f.setFlightNumber(updated.getFlightNumber());
        }
        if (updated.getOrigin() != null) f.setOrigin(updated.getOrigin());
        if (updated.getDestination() != null)
f.setDestination(updated.getDestination());
        if (updated.getDepartureTime() != null)
f.setDepartureTime(updated.getDepartureTime());
        if (updated.getSeatsAvailable() != null) {
            if (updated.getSeatsAvailable() < 0) throw new
IllegalArgumentException("Seats cannot be negative");
            f.setSeatsAvailable(updated.getSeatsAvailable());
        }
        return flightRepository.save(f);
    }

    public void deleteFlight(Long id) {
        Flight f = getFlightById(id);
        flightRepository.delete(f);
    }

    @Transactional
```

```
    public void adjustSeats(Long flightId, int delta) {
        Flight f = getFlightById(flightId);
        int newSeats = f.getSeatsAvailable() + delta;
        if (newSeats < 0) throw new IllegalArgumentException("Flight cannot have
negative seats");
        f.setSeatsAvailable(newSeats);
        flightRepository.save(f);
    }
}
src/main/java/com/example/flight/service/ReservationService.java
java
Copy
Edit
package com.example.flight.service;

import com.example.flight.entity.Flight;
import com.example.flight.entity.Reservation;
import com.example.flight.exception.FlightNotFoundException;
import com.example.flight.exception.NotEnoughSeatsException;
import com.example.flight.repository.FlightRepository;
import com.example.flight.repository.ReservationRepository;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

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;
    }

    @Transactional
    public Reservation makeReservation(Long flightId, String passengerName, String
passengerEmail, int seats) {
        Flight flight = flightRepository.findById(flightId)
                .orElseThrow(() -> new FlightNotFoundException("Flight not found: "
+ flightId));

        if (seats <= 0) throw new IllegalArgumentException("Seats booked must be >
0");
        if (flight.getSeatsAvailable() == null) throw new
IllegalArgumentException("Flight seats availability not set");
        if (flight.getSeatsAvailable() < seats) {
```

```
            throw new NotEnoughSeatsException("Not enough seats available.
Requested: " + seats + ", available: " + flight.getSeatsAvailable());
        }

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

        Reservation reservation = new Reservation(passengerName, passengerEmail,
seats, LocalDateTime.now(), flight);
        return reservationRepository.save(reservation);
    }

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

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

    @Transactional
    public void cancelReservation(Long reservationId) {
        Reservation res = reservationRepository.findById(reservationId)
                .orElseThrow(() -> new IllegalArgumentException("Reservation not
found: " + reservationId));
        Flight flight = res.getFlight();
        if (flight == null) throw new IllegalArgumentException("Associated flight
missing for reservation: " + reservationId);

        int newSeats = (flight.getSeatsAvailable() == null ? 0 :
flight.getSeatsAvailable()) + res.getSeatsBooked();
        if (newSeats < 0) throw new IllegalArgumentException("Invalid seat restore
operation");
        flight.setSeatsAvailable(newSeats);
        flightRepository.save(flight);
        reservationRepository.delete(res);
    }
}
8 — Controllers (REST endpoints)
src/main/java/com/example/flight/controller/FlightController.java
java
Copy
Edit
package com.example.flight.controller;

import com.example.flight.entity.Flight;
import com.example.flight.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> createFlight(@RequestBody Flight flight) {
        Flight created = flightService.createFlight(flight);
        return ResponseEntity.status(201).body(created);
    }

    @GetMapping
    public List<Flight> getAll() { return flightService.getAllFlights(); }

    @GetMapping("/{id}")
    public Flight getById(@PathVariable Long id) { return
flightService.getFlightById(id); }

    @PutMapping("/{id}")
    public Flight update(@PathVariable Long id, @RequestBody Flight flight) {
return flightService.updateFlight(id, flight); }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> delete(@PathVariable Long id) {
        flightService.deleteFlight(id);
        return ResponseEntity.noContent().build();
    }
}
src/main/java/com/example/flight/controller/ReservationController.java
java
Copy
Edit
package com.example.flight.controller;

import com.example.flight.entity.Reservation;
import com.example.flight.service.ReservationService;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Map;

@RestController
@RequestMapping("/api/reservations")
public class ReservationController {
    private final ReservationService reservationService;
```

```
    public ReservationController(ReservationService reservationService) {
this.reservationService = reservationService; }

    // Body expects: { "flightId": 1, "passengerName": "John", "passengerEmail":
"a@b.com", "seatsBooked": 2 }
    @PostMapping
    public ResponseEntity<Reservation> makeReservation(@RequestBody Map<String,
Object> payload) {
        Long flightId = Long.valueOf(payload.get("flightId").toString());
        String name = payload.get("passengerName").toString();
        String email = payload.get("passengerEmail").toString();
        int seats = Integer.parseInt(payload.get("seatsBooked").toString());
        Reservation res = reservationService.makeReservation(flightId, name, email,
seats);
        return ResponseEntity.status(201).body(res);
    }

    @GetMapping
    public List<Reservation> getAll() { return
reservationService.getAllReservations(); }

    @GetMapping("/flight/{flightId}")
    public List<Reservation> getByFlight(@PathVariable Long flightId) { return
reservationService.getReservationsByFlight(flightId); }

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

9 — How to run

From project root:

```
arduino
Copy
Edit
mvn spring-boot:run
```

Access:

Swagger UI: http://localhost:8080/swagger-ui.html

H2 console: http://localhost:8080/h2-console (JDBC URL: jdbc:h2:mem:flightsdb, user sa, no password)

10 — Two terminal examples (curl) — create flight & make reservation

Example 1 — Create a flight

```
bash
Copy
```

```
Edit
curl -X POST http://localhost:8080/api/flights \
  -H "Content-Type: application/json" \
  -d '{
    "flightNumber": "AI101",
    "origin": "Bengaluru",
    "destination": "Mumbai",
    "departureTime": "2025-09-01T10:00:00",
    "seatsAvailable": 150
  }'
Expected: JSON response for created flight (status 201).

Example 2 — Make a reservation for that flight (flightId = 1)

bash
Copy
Edit
curl -X POST http://localhost:8080/api/reservations \
  -H "Content-Type: application/json" \
  -d '{
    "flightId": 1,
    "passengerName": "John Doe",
    "passengerEmail": "john@example.com",
    "seatsBooked": 2
  }'
```