

✅ Case Study: Product-Order Management System (With Mockito Testing)

🧩 Objective Develop a simple Product-Order system using Spring Boot with MySQL. Test the business logic of services using Mockito. No integration testing or H2 database involved.

📄 Functional Requirements

1. 2. 3. 4. Admin can add, view, and update products. Users can place orders for available products. The system reduces stock when an order is placed. Each order stores order details and is linked to the product.

🧱 Entity Design

1. Product

- productId (PK)
- name
- price
- availableQuantity

2. Order

- orderId (PK)
- product (ManyToOne)
- orderDate
- quantityOrdered

🧬 Repository Layer

- ProductRepository extends JpaRepository
- OrderRepository extends JpaRepository


⚙️ Service Layer ProductService

- addProduct(Product p)
- getAllProducts()
- updateStock(Long productId, int qty) OrderService
- placeOrder(Long productId, int quantity)
 - Check if stock is available

- Create order
- Reduce product quantity

Controller Layer /api/products

- POST / → Add product
- GET / → List all products
- PUT /{id}/stock → Update stock /api/orders
- POST / → Place order
- GET / → List all orders


 Unit Testing Strategy (Mockito only) We test only the service layer using Mockito, without real DB access.

ProductServiceTest

- Mock ProductRepository
- Test:
 - Adding product
 - Fetching all products
 - Stock update logic

OrderServiceTest

- Mock OrderRepository and ProductRepository
- Test:
 - Order placed successfully when stock is available
 - Order fails if stock is insufficient

 Database Setup (MySQL) In your application.properties:

spring.datasource.url=jdbc:mysql://localhost:3306/ product_order_db

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update

No need for test profiles or alternate configurations.

Tools & Tech Stack

- Spring Boot 3+
- Spring Data JPA
- MySQL
- JUnit 5
- Mockito

Summary of Benefits

- Clean separation of concerns (MVC + layered architecture)
- Business logic isolated for testing
- Mockito ensures fast, DB-independent testing
- MySQL used consistently in development: and testing

Entity class:

Order.java:

```
package com.example.productordersystem.entity;
```

```
import java.time.LocalDate;
```

```
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 = "orders")
```

```
public class Order {
```

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

private Long orderId;

@ManyToOne

@JoinColumn(name = "product_id")

private Product product;

private LocalDate orderDate;

private int quantityOrdered;

public Long getOrderId() {

return orderId;

}

public Product getProduct() {

return product;

}

public LocalDate getOrderDate() {

return orderDate;

}

public int getQuantityOrdered() {

return quantityOrdered;

}

public void setOrderId(Long orderId) {

this.orderId = orderId;

}

public void setProduct(Product product) {

this.product = product;

```

    }

    public void setOrderDate(LocalDate orderDate) {

        this.orderDate = orderDate;

    }

    public void setQuantityOrdered(int quantityOrdered) {

        this.quantityOrdered = quantityOrdered;

    }

    public Order() {

    }

}

```

Product.java:

```

package com.example.productordersystem.entity;

```

```

import jakarta.persistence.Entity;

```

```

import jakarta.persistence.GeneratedValue;

```

```

import jakarta.persistence.GenerationType;

```

```

import jakarta.persistence.Id;

```

```

@Entity

```

```

public class Product {

```

```

    @Id

```

```

    @GeneratedValue(strategy = GenerationType.IDENTITY)

```

```
private Long productId;

private String name;

private double price;

private int availableQuantity;

public Long getProductId() {

    return productId;

}

public String getName() {

    return name;

}

public double getPrice() {

    return price;

}

public int getAvailableQuantity() {

    return availableQuantity;

}

public void setProductId(Long productId) {

    this.productId = productId;

}

public void setName(String name) {

    this.name = name;

}

public void setPrice(double price) {

    this.price = price;

}

public void setAvailableQuantity(int availableQuantity) {

    this.availableQuantity = availableQuantity;

}
```

```
}
```

```
public Product() {
```

```
}
```

```
}
```

Repository:

OrderRepository:

```
package com.example.productordersystem.repository;
```

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

```
import com.example.productordersystem.entity.Order;
```

```
public interface OrderRepository extends JpaRepository<Order, Long> {
```

```
}
```

ProductRepository:

```
package com.example.productordersystem.repository;
```

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

```
import com.example.productordersystem.entity.Product;
```

```
public interface ProductRepository extends JpaRepository<Product, Long> {
```

```
}
```

Service class:

OrderService:

```
package com.example.productordersystem.service;
```

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

```
import com.example.productordersystem.entity.Order;
```

```
public interface OrderService {  
  
    Order placeOrder(Long productId, int quantity);  
  
    List<Order> getAllOrders();  
  
}
```

ProductService:

```
package com.example.productordersystem.service;
```

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

```
import com.example.productordersystem.entity.Product;
```

```
public interface ProductService {  
  
    List<Product> getAllProducts();  
  
    void updateStock(Long productId, int qty);  
  
    Product addProduct(Product product);  
  
}
```


IMPL CLASS:

OrderServiceImpl:

package com.example.productordersystem.service.impl;

import java.time.LocalDate;

import java.util.List;

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

import org.springframework.stereotype.Service;

import com.example.productordersystem.entity.Order;

import com.example.productordersystem.entity.Product;

import com.example.productordersystem.repository.OrderRepository;

import com.example.productordersystem.repository.ProductRepository;

import com.example.productordersystem.service.OrderService;

@Service

public class OrderServiceImpl **implements** OrderService{

@Autowired

private ProductRepository productRepo;

@Autowired

private OrderRepository orderRepo;

@Override

public Order placeOrder(Long productId, **int** quantity) {

Product product = productRepo.findById(productId)

.orElseThrow(() -> **new** RuntimeException("Product not found"));

if (product.getAvailableQuantity() < quantity) {

```
        throw new RuntimeException("Insufficient stock");
    }
}
```

```
Order order = new Order();

order.setProduct(product);

order.setOrderDate(LocalDate.now());

order.setQuantityOrdered(quantity);

orderRepo.save(order);

product.setAvailableQuantity(product.getAvailableQuantity() - quantity);

productRepo.save(product);

return order;
}
```

```
@Override
```

```
public List<Order> getAllOrders() {

    return orderRepo.findAll();

}

}
```

ProductServiceImpl:

```
package com.example.productordersystem.service.impl;
```

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

```
import java.util.Optional;
```

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

```
import com.example.productordersystem.entity.Product;
```

```
import com.example.productordersystem.repository.ProductRepository;
```

```
import com.example.productordersystem.service.ProductService;
```

```
@Service
```

```
public class ProductServiceImpl implements ProductService {
```

```
    private final ProductRepository productRepo;
```

```
    public ProductServiceImpl(ProductRepository productRepo) {
```

```
        this.productRepo = productRepo;
```

```
    }
```

```
@Override
```

```
    public Product addProduct(Product product) {
```

```
        return productRepo.save(product);
```

```
    }
```

```
@Override
```

```
    public List<Product> getAllProducts() {
```

```
        return productRepo.findAll();
```

```
    }
```

```
@Override
```

```
    public void updateStock(Long productId, int newQty) {
```

```
        Optional<Product> optionalProduct = productRepo.findById(productId);
```

```
        if (optionalProduct.isPresent()) {
```

```
            Product product = optionalProduct.get();
```

```
            product.setAvailableQuantity(newQty);
```

```
            productRepo.save(product);
```

```
        } else {
```

```
            throw new RuntimeException("Product not found with ID: " + productId);
```

```
}  
  
}  
  
}
```

Controller class:

OrderController:

```
package com.example.productordersystem.controller;  
  
import java.util.List;  
  
import org.springframework.beans.factory.annotation.Autowired;  
  
import org.springframework.http.HttpStatus;  
  
import org.springframework.http.ResponseEntity;  
  
import org.springframework.web.bind.annotation.GetMapping;  
  
import org.springframework.web.bind.annotation.PostMapping;  
  
import org.springframework.web.bind.annotation.RequestMapping;  
  
import org.springframework.web.bind.annotation.RequestParam;  
  
import org.springframework.web.bind.annotation.RestController;  
  
  
import com.example.productordersystem.entity.Order;  
  
import com.example.productordersystem.service.OrderService;  
  
  
@RestController  
  
@RequestMapping("/api/orders")  
  
public class OrderController {  
  
    @Autowired  
  
    private OrderService orderService;
```

@PostMapping

```
public ResponseEntity<Order> placeOrder(@RequestParam Long productId, @RequestParam int quantity) {  
  
    return new ResponseEntity<>(orderService.placeOrder(productId, quantity), HttpStatus.CREATED);  
  
}
```

@GetMapping

```
public List<Order> getAll() {  
  
    return orderService.getAllOrders();  
  
}  
  
}
```

ProductController:

```
package com.example.productordersystem.controller;
```

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

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

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

```
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.PathVariable;
```

```
import org.springframework.web.bind.annotation.PostMapping;
```

```
import org.springframework.web.bind.annotation.PutMapping;
```

```
import org.springframework.web.bind.annotation.RequestBody;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestParam;
```

```
import org.springframework.web.bind.annotation.RestController;
```

```
import com.example.productordersystem.entity.Product;
```

```
import com.example.productordersystem.service.ProductService;
```

```
@RestController
```

```
@RequestMapping("/api/products")
```

```
public class ProductController {
```

```
    @Autowired
```

```
    private ProductService productService;
```

```
    @PostMapping
```

```
    public ResponseEntity<Product> add(@RequestBody Product product) {
```

```
        return new ResponseEntity<>(productService.addProduct(product), HttpStatus.CREATED);
```

```
    }
```

```
    @GetMapping
```

```
    public List<Product> getAll() {
```

```
        return productService.getAllProducts();
```

```
    }
```

```
    @PutMapping("/{id}/stock")
```

```
    public ResponseEntity<Void> updateStock(@PathVariable Long id, @RequestParam int qty) {
```

```
        productService.updateStock(id, qty);
```

```
        return ResponseEntity.ok().build();
```

```
    }
```

```
}
```

MainClass:

ProductOrderSystemApplication:

```
package com.example.productordersystem;

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

@SpringBootApplication

public class ProductOrderSystemApplication {

    public static void main(String[] args) {

        SpringApplication.run(ProductOrderSystemApplication.class, args);

    }

}
```

application.properties:

```
spring.application.name=ProductOrderSystem

spring.datasource.url=jdbc:mysql://localhost:3306/product_order_db
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

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/>
```

```
</parent>
```

```
<groupId>com.example</groupId>
```

```
<artifactId>productordersystem</artifactId>
```

```
<version>0.0.1-SNAPSHOT</version>
```

```
<name>ProductOrderSystem</name>
```

```
<description>Spring Boot Product-Order Management System</description>
```

```
<properties>
```

```
    <java.version>17</java.version>
```

```
</properties>
```

```
<dependencies>
```

```
    <!-- Spring Boot Web Starter -->
```

```
    <dependency>
```

```
        <groupId>org.springframework.boot</groupId>
```

```
        <artifactId>spring-boot-starter-web</artifactId>
```

```
    </dependency>
```

```
    <!-- Spring Data JPA -->
```



```
<dependency>

  <groupId>org.springframework.boot</groupId>

  <artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>
```

```
<!-- MySQL JDBC Driver -->
```

```
<dependency>

  <groupId>com.mysql</groupId>

  <artifactId>mysql-connector-j</artifactId>

  <scope>runtime</scope>

</dependency>
```

```
<!-- Lombok (For reducing boilerplate) -->
```

```
<dependency>

  <groupId>org.projectlombok</groupId>

  <artifactId>lombok</artifactId>

  <optional>true</optional>

</dependency>
```

```
<!-- Devtools (for hot reload) -->
```

```
<dependency>

  <groupId>org.springframework.boot</groupId>

  <artifactId>spring-boot-devtools</artifactId>

  <scope>runtime</scope>

  <optional>true</optional>

</dependency>
```

```
<!-- JUnit 5 + Mockito for Unit Testing -->
```

```
<dependency>
```

`<groupId>org.springframework.boot</groupId>`

`<artifactId>spring-boot-starter-test</artifactId>`

`<scope>test</scope>`

`<exclusions>`

`<!-- Exclude unwanted engine if you're using only JUnit 5 -->`

`<exclusion>`

`<groupId>org.junit.vintage</groupId>`

`<artifactId>junit-vintage-engine</artifactId>`

`</exclusion>`

`</exclusions>`

`</dependency>`

`<!-- Mockito Core (optional but useful for clarity) -->`

`<dependency>`

`<groupId>org.mockito</groupId>`

`<artifactId>mockito-core</artifactId>`

`<scope>test</scope>`

`</dependency>`

`</dependencies>`

`<build>`

`<plugins>`

`<!-- Java Compilation Plugin -->`

`<plugin>`

`<groupId>org.apache.maven.plugins</groupId>`

`<artifactId>maven-compiler-plugin</artifactId>`

`<version>3.11.0</version>`

<configuration>

<source>\${java.version}</source>

<target>\${java.version}</target>

<annotationProcessorPaths>

<path>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.32</version>

</path>

</annotationProcessorPaths>

</configuration>

</plugin>

<!-- Spring Boot Maven Plugin -->

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<version>3.5.4</version>

</plugin>

</plugins>

</build>

</project>