# Product-Order Management System

## SpringtestCasestudyApplication:

```
package com.example.springtest_casestudy;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringtestCasestudyApplication {
	public static void main(String[] args) {
		SpringApplication.run(SpringtestCasestudyApplication.class, args);
	}
}
```

## Product.java:

```
package com.example.springtest_casestudy.entity;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
@Entity
@Table(name="`product`")
public class Product {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;
```

```
    private String name;
    private double price;
    private int availableQuantity;
    public Product() {}
    public Product(Long id, String name, double price, int
availableQuantity){
        this.id = id;
        this.name = name;
        this.price = price;
        this.availableQuantity = availableQuantity;
    }
    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 double getPrice() {
        return price;
    }
    public void setPrice(double price) {
```

```
            this.price = price;
        }
        public int getAvailableQuantity() {
            return availableQuantity;
        }
        public void setAvailableQuantity(int availableQuantity) {
            this.availableQuantity = availableQuantity;
        }
}
```

## Order.java:

```
package com.example.springtest_casestudy.entity;
import java.time.LocalDate;
import jakarta.persistence.*;
@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 Order() {}
```

```
    public Order(Long orderId, Product product, LocalDate orderDate, int
quantityOrdered) {
        this.orderId = orderId;
        this.product = product;
        this.orderDate = orderDate;
        this.quantityOrdered = quantityOrdered;
    }
    public Long getOrderId() {
        return orderId;
    }
    public void setOrderId(Long orderId) {
        this.orderId = orderId;
    }
    public Product getProduct() {
        return product;
    }
    public void setProduct(Product product) {
        this.product = product;
    }
    public LocalDate getOrderDate() {
        return orderDate;
    }
    public void setOrderDate(LocalDate orderDate) {
        this.orderDate = orderDate;
    }
    public int getQuantityOrdered() {
        return quantityOrdered;
```

```
    }
    public void setQuantityOrdered(int quantityOrdered) {
        this.quantityOrdered = quantityOrdered;
    }
}
```

## ProductRepository.java:

```
package com.example.springtest_casestudy.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.springtest_casestudy.entity.Product;
public interface ProductRepository extends JpaRepository<Product, Long> {
}
```

## OrderRepository.java:

```
package com.example.springtest_casestudy.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.springtest_casestudy.entity.Order;
public interface OrderRepository extends JpaRepository<Order, Long> {
}
```

## ProductService:

```
package com.example.springtest_casestudy.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.springtest_casestudy.entity.Product;
import com.example.springtest_casestudy.repository.ProductRepository;
@Service
```

```
public class ProductService {
        @Autowired
         private ProductRepository productRepo;

         public Product addProduct(Product product) {
             return productRepo.save(product);
          }
          public List<Product> getAllProducts() {
             return productRepo.findAll();
          }
          public void updateStock(Long productId, int qty) {
             Product product = productRepo.findById(productId)
                    .orElseThrow(() -> new RuntimeException("Product not
found"));
             product.setAvailableQuantity(qty);
             productRepo.save(product);
          }

}
```

## OrderService:

```
package com.example.springtest_casestudy.service;
import java.time.LocalDate;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.springtest_casestudy.entity.*;
```

```
import com.example.springtest_casestudy.repository.*;
@Service
public class OrderService {
    @Autowired
    private OrderRepository orderRepo;
    @Autowired
    private ProductRepository productRepo;
    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 available");
        }
        product.setAvailableQuantity(product.getAvailableQuantity() - quantity);
        productRepo.save(product);
        Order order = new Order();
        order.setProduct(product);
        order.setQuantityOrdered(quantity);
        order.setOrderDate(LocalDate.now());
        return orderRepo.save(order);
    }
    public List<Order> getAllOrders() {
        return orderRepo.findAll();
    }
}
```

## ProductController:

```
package com.example.springtest_casestudy.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import com.example.springtest_casestudy.entity.Product;
import com.example.springtest_casestudy.service.ProductService;
import java.util.List;

@RestController
@RequestMapping("/api/products")
public class ProductController {
    @Autowired
    private ProductService productService;
    @PostMapping("/")
    public Product addProduct(@RequestBody Product product) {
        return productService.addProduct(product);
    }
    @GetMapping("/")
    public List<Product> getAllProducts() {
        return productService.getAllProducts();
    }
    @PutMapping("/{id}/stock")
    public void updateStock(@PathVariable Long id, @RequestParam int qty) {
        productService.updateStock(id, qty);
    }
}
```

## OrderController:

```
package com.example.springtest_casestudy.controller;

import java.util.List;

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

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

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

import com.example.springtest_casestudy.entity.Order;

import com.example.springtest_casestudy.service.OrderService;

@RestController

@RequestMapping("/api/orders")

public class OrderController {

    @Autowired

    private OrderService orderService;

    @PostMapping("/")

    public Order placeOrder(@RequestParam Long productId, @RequestParam
int quantity) {

        return orderService.placeOrder(productId, quantity);

    }

    @GetMapping("/")

    public List<Order> getAllOrders() {

        return orderService.getAllOrders();

    }

}
```

## ProductServiceTest:

```
package com.example.springtest_casestudy;

import org.junit.jupiter.api.Test;
```

```
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;
import com.example.springtest_casestudy.entity.Product;
import com.example.springtest_casestudy.repository.ProductRepository;
import com.example.springtest_casestudy.service.ProductService;
import java.util.*;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;

public class ProductServiceTest {
        @Mock
    private ProductRepository productRepo;

    @InjectMocks
    private ProductService productService;

    public ProductServiceTest() {
        MockitoAnnotations.openMocks(this);
    }
    @Test
    void testAddProduct() {
        Product p = new Product(null,"Test Product", 99.0, 10);
        when(productRepo.save(p)).thenReturn(p);
        assertEquals(p, productService.addProduct(p));
    }
```

```
    @Test
    void testUpdateStock() {
        Product p = new Product(null,"Item", 10.0, 5);
        when(productRepo.findById(1L)).thenReturn(Optional.of(p));
        productService.updateStock(1L, 20);
        verify(productRepo, times(1)).save(p);
        assertEquals(20, p.getAvailableQuantity());
    }
}
```

## OrderServiceTest:

```
package com.example.springtest_casestudy;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;
import static org.mockito.ArgumentMatchers.any;
import static org.mockito.Mockito.when;

import java.util.Optional;

import org.junit.jupiter.api.Test;
import org.mockito.*;

import com.example.springtest_casestudy.entity.Order;
import com.example.springtest_casestudy.entity.Product;
import com.example.springtest_casestudy.repository.OrderRepository;
import com.example.springtest_casestudy.repository.ProductRepository;
```

```
import com.example.springtest_casestudy.service.OrderService;

public class OrderServiceTest {

        @Mock
    private ProductRepository productRepo;

    @Mock
    private OrderRepository orderRepo;

    @InjectMocks
    private OrderService orderService;

    public OrderServiceTest() {
       MockitoAnnotations.openMocks(this);
    }

    @Test
    void testPlaceOrderSuccess() {
       Product p = new Product(null, "Laptop", 100.0, 10);
       when(productRepo.findById(1L)).thenReturn(Optional.of(p));
       when(orderRepo.save(any(Order.class))).thenAnswer(inv ->
inv.getArguments()[0]);

       Order order = orderService.placeOrder(1L, 5);
       assertEquals(5, order.getQuantityOrdered());
       assertEquals(5, p.getAvailableQuantity());
```

```
    }
    @Test
    void testPlaceOrderFailDueToStock() {
        Product p = new Product(null,"LowStock", 20.0, 2);
        when(productRepo.findById(1L)).thenReturn(Optional.of(p));
        assertThrows(RuntimeException.class, () ->
orderService.placeOrder(1L, 5));
    }
}
```