Product-Order Management System (With Mockito Testing) case Study

Project Folder Structure

```
product-order-management/
 ├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/example/productorder/
│   │   │       ├── ProductOrderApplication.java
│   │   │
│   │   │       ├── controller/
│   │   │       │   ├── ProductController.java
│   │   │       │   └── OrderController.java
│   │   │
│   │   │       ├── entity/
│   │   │       │   ├── Product.java
│   │   │       │   └── Order.java
│   │   │
│   │   │       ├── repository/
│   │   │       │   ├── ProductRepository.java
│   │   │       │   └── OrderRepository.java
│   │   │
│   │   │       ├── service/
│   │   │       │   ├── ProductService.java
│   │   │       │    ├── OrderService.java
│   │   │       │
│   │   │       └── serviceimpl
│   │   │            ├── ProductServiceImpl.java
│   │   │            └── OrderServiceImpl.java
│   │
│   │   └── resources/
│   │        ├── application.properties
│   │
│   │
│
│
├── test/
│   └── java/
│       └── com/example/productorder/
│            ├── service/
│            │    ├── ProductServiceTest.java
│            │    └── OrderServiceTest.java
│
├── pom.xml
```

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
```

```xml
                        http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>product-order-management</artifactId>
    <version>1.0.0</version>
    <packaging>jar</packaging>

    <name>Product Order Management</name>
    <description>Product-Order Management System with Mockito Testing</description>

    <properties>
        <java.version>17</java.version>
        <spring.boot.version>3.1.0</spring.boot.version>
    </properties>

    <dependencies>
        <!-- Spring Boot Starter Web -->
        <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 Driver -->
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>

<!-- Lombok (Optional but recommended) -->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>

<!-- JUnit 5 for Testing -->
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <scope>test</scope>
</dependency>

<!-- Mockito for Unit Testing -->
<dependency>
    <groupId>org.mockito</groupId>
```

```xml
        <artifactId>mockito-core</artifactId>
        <version>5.7.0</version>
        <scope>test</scope>
    </dependency>

    <!-- Mockito Extension (JUnit 5) -->
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-junit-jupiter</artifactId>
        <version>5.7.0</version>
        <scope>test</scope>
    </dependency>

    <!-- Spring Boot Test (optional, used for annotations like @WebMvcTest) -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
        <exclusions>
            <exclusion>
                <groupId>org.junit.vintage</groupId>
                <artifactId>junit-vintage-engine</artifactId>
            </exclusion>
        </exclusions>
    </dependency>

</dependencies>
```

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

            <!-- Maven Compiler Plugin -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.11.0</version>
                <configuration>
                    <source>17</source>
                    <target>17</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

1.src/main/java
com.example.productorder
ProductOrderApplication.java(Main Class)

```
package com.example.productorder;
```

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

com.example.controller
ProductController.java

```
package com.example.controller;
import com.example.productorder.entity.Product;
import com.example.productorder.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
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> getAll() {
        return productService.getAllProducts();
    }

    @PutMapping("/{id}/stock")
    public Product updateStock(@PathVariable Long id, @RequestParam int quantity) {
        return productService.updateStock(id, quantity);
    }
}
```

OrderController.java

```
package com.example.controller;
import com.example.productorder.entity.Order;
import com.example.productorder.service.OrderService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@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();
    }
}
```

Com.example.entity

Product.java

```
package com.example.entity;
import jakarta.persistence.*;
@Entity
public class Product {

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

    private String name;
    private Double price;
    private Integer availableQuantity;
}
```

Order.java

```
package com.example.entity;
import jakarta.persistence.*;
import java.time.LocalDateTime;
@Entity
@Table(name = "product_order")
public class Order {

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

    @ManyToOne
    private Product product;

    private LocalDateTime orderDate;

    private Integer quantityOrdered;

    // Getters and Setters
}
```

Com.example.repository

ProductRepository.java

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

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

OrderRepository.java

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

com.example.service

ProductService.java

```
package com.example.service;
import com.example.entity.Product;
import java.util.List;


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

OrderService.java

```
package com.example.service;
import com.example.entity.Order;
import java.util.List;
public interface OrderService {
    Order placeOrder(Long productId, int quantity);
    List<Order> getAllOrders();
}
```

com.example.serviceimpl

ProductServiceImpl

```
package com.example.service.impl;
import com.example.entity.Product;
import com.example.repository.ProductRepository;
import com.example.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class ProductServiceImpl implements ProductService {

    @Autowired
    private ProductRepository productRepository;

    public Product addProduct(Product product) {
        return productRepository.save(product);
    }

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

2.src/test/java
com.example.productserviceorder

ProductServiceTest

```
@ExtendWith(MockitoExtension.class)
public class ProductServiceTest {

    @Mock
    private ProductRepository productRepository;

    @InjectMocks
    private ProductServiceImpl productService;

    @Test
    void testAddProduct() {
        Product p = new Product();
        p.setName("Keyboard");
        when(productRepository.save(any())).thenReturn(p);
        Product result = productService.addProduct(p);
        assertEquals("Keyboard", result.getName());
    }

    @Test
    void testGetAllProducts() {
        when(productRepository.findAll()).thenReturn(List.of(new Product()));
        assertEquals(1, productService.getAllProducts().size());
    }

    @Test
    void testUpdateStock() {
        Product p = new Product();
        p.setAvailableQuantity(10);
        when(productRepository.findById(1L)).thenReturn(Optional.of(p));
        when(productRepository.save(any())).thenReturn(p);

        Product updated = productService.updateStock(1L, 20);
        assertEquals(20, updated.getAvailableQuantity());
    }
}
```

OrderServiceTest

```
@ExtendWith(MockitoExtension.class)
public class OrderServiceTest {

    @Mock
```

```
private OrderRepository orderRepository;

@Mock
private ProductRepository productRepository;

@InjectMocks
private OrderServiceImpl orderService;

@Test
void testPlaceOrderSuccess() {
    Product p = new Product();
    p.setProductId(1L);
    p.setAvailableQuantity(10);

    when(productRepository.findById(1L)).thenReturn(Optional.of(p));
    when(productRepository.save(any())).thenReturn(p);
    when(orderRepository.save(any())).thenAnswer(inv -> inv.getArgument(0));

    Order order = orderService.placeOrder(1L, 5);
    assertEquals(5, order.getQuantityOrdered());
}

@Test
void testPlaceOrderFailure() {
    Product p = new Product();
    p.setAvailableQuantity(3);
```

```
        when(productRepository.findById(1L)).thenReturn(Optional.of(p));


        assertThrows(RuntimeException.class, () -> {
            orderService.placeOrder(1L, 5);
        });
    }
}
```