Day-13 Assignment

Order.java

```
package com.example.productorder.entity;
import jakarta.persistence.*;
import java.time.LocalDate;
@Entity(name = "orders")
public class Order {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long orderId;
   @ManyToOne
   private Product product;
   private LocalDate orderDate;
   private int 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;
        }
}
```

Product.java

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

import jakarta.persistence.*;

@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 void setProductId(Long productId) {
            this.productId = productId;
        }
        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;
        }

    // Getters and Setters
}
```

OrderController.java

```
package com.example.productorder.controller;
import com.example.productorder.entity.Order;
import com.example.productorder.service.OrderService;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@RequestMapping("/api/orders")
public class OrderController {
    private final OrderService service;
    public OrderController(OrderService service) {
        this.service = service;
```

```
    }
    @PostMapping
    public Order placeOrder(@RequestParam Long productId, @RequestParam int quantity)
{
        return service.placeOrder(productId, quantity);
    }
    @GetMapping
    public List<Order> getAllOrders() {
        return service.getAllOrders();
    }
}
```

ProductController.java

```
package com.example.productorder.controller;
import com.example.productorder.entity.Product;
import com.example.productorder.service.ProductService;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@RequestMapping("/api/products")
public class ProductController {
    private final ProductService service;
    public ProductController(ProductService service) {
        this.service = service;
    }
    @PostMapping
    public Product addProduct(@RequestBody Product product) {
        return service.addProduct(product);
    }
    @GetMapping
    public List<Product> getAll() {
        return service.getAllProducts();
    }
    @PutMapping("/{id}/stock")
    public Product updateStock(@PathVariable Long id, @RequestParam int qty) {
        return service.updateStock(id, qty);
    }
}
```

OrderRepository interface

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

}
```

PoductRepository interface

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

import com.example.productorder.entity.Product;
import org.springframework.data.jpa.repository.JpaRepository;

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

OrderService.java

```
package com.example.productorder.service;
import com.example.productorder.entity.Order;
import com.example.productorder.entity.Product;
import com.example.productorder.repository.OrderRepository;
import com.example.productorder.repository.ProductRepository;
import org.springframework.stereotype.Service;
import java.time.LocalDate;
import java.util.List;
@Service
public class OrderService {
    private final OrderRepository orderRepo;
    private final ProductRepository productRepo;
    public OrderService(OrderRepository orderRepo, ProductRepository productRepo) {
        this.orderRepo = orderRepo;
        this.productRepo = productRepo;
    }
    public Order placeOrder(Long productId, int quantity) {
        Product product = productRepo.findById(productId).orElseThrow();
        if (product.getAvailableQuantity() < quantity) {
            throw new RuntimeException("Not enough stock");
        }
        product.setAvailableQuantity(product.getAvailableQuantity() - quantity);
        productRepo.save(product);
        Order order = new Order();
        order.setProduct(product);
        order.setOrderDate(LocalDate.now());
        order.setQuantityOrdered(quantity);
        return orderRepo.save(order);
    }
    public List<Order> getAllOrders() {
        return orderRepo.findAll();
    }
}
```

ProductService.java

```
package com.example.productorder.service;
import com.example.productorder.entity.Product;
```

```
import com.example.productorder.repository.ProductRepository;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class ProductService {
    private final ProductRepository repo;
    public ProductService(ProductRepository repo) {
        this.repo = repo;
    }
    public Product addProduct(Product product) {
        return repo.save(product);
    }
    public List<Product> getAllProducts() {
        return repo.findAll();
    }
    public Product updateStock(Long productId, int qty) {
        Product p = repo.findById(productId).orElseThrow();
        p.setAvailableQuantity(qty);
        return repo.save(p);
    }
}
```

Application.properties

```
spring.application.name=productordersystem
spring.datasource.url=jdbc:mysql://localhost:3306/product_order_db
spring.datasource.username=root
spring.datasource.password=Sravani@123
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
server.port=8080
```

OrderServiceTest.java

```
package com.example.productorder.service;
import com.example.productorder.entity.Order;
import com.example.productorder.entity.Product;
import com.example.productorder.repository.OrderRepository;
import com.example.productorder.repository.ProductRepository;
import org.junit.jupiter.api.Test;
import java.util.Optional;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;
class OrderServiceTest {
    private final ProductRepository productRepo = mock(ProductRepository.class);
    private final OrderRepository orderRepo = mock(OrderRepository.class);
    private final OrderService service = new OrderService(orderRepo, productRepo);
    @Test
```

```
    void testPlaceOrderSuccess() {
        Product product = new Product();
        product.setProductId(1L);
        product.setAvailableQuantity(10);
        when(productRepo.findById(1L)).thenReturn(Optional.of(product));
        when(orderRepo.save(any())).thenReturn(new Order());
        Order order = service.placeOrder(1L, 5);
        verify(productRepo).save(product);
        assertNotNull(order);
    }
    @Test
    void testPlaceOrderFailsDueToInsufficientStock() {
        Product product = new Product();
        product.setAvailableQuantity(2);
        when(productRepo.findById(1L)).thenReturn(Optional.of(product));
        RuntimeException ex = assertThrows(RuntimeException.class, () -> {
            service.placeOrder(1L, 5);
        });
        assertEquals("Not enough stock", ex.getMessage());
    }
}
```

ProductServiceTest.java

```
package com.example.productorder.service;
import com.example.productorder.entity.Product;
import com.example.productorder.repository.ProductRepository;
import org.junit.jupiter.api.Test;
import java.util.List;
import java.util.Optional;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;
class ProductServiceTest {
    private final ProductRepository repo = mock(ProductRepository.class);
    private final ProductService service = new ProductService(repo);
    @Test
    void testAddProduct() {
        Product p = new Product();
        p.setName("Phone");
        when(repo.save(p)).thenReturn(p);
        Product result = service.addProduct(p);
        assertEquals("Phone", result.getName());
    }
    @Test
    void testGetAllProducts() {
        when(repo.findAll()).thenReturn(List.of(new Product()));
        assertEquals(1, service.getAllProducts().size());
    }
    @Test
```

```
    void testUpdateStock() {
        Product p = new Product();
        p.setAvailableQuantity(10);
        when(repo.findById(1L)).thenReturn(Optional.of(p));
        when(repo.save(any())).thenReturn(p);
        Product updated = service.updateStock(1L, 20);
        assertEquals(20, updated.getAvailableQuantity());
    }
}
```

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.example</groupId>
        <artifactId>productordersystem</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>productordersystem</name>
        <description>Demo project for Spring Test</description>
        <url/>
        <licenses>
                <license/>
        </licenses>
        <developers>
                <developer/>
        </developers>
        <scm>
                <connection/>
                <developerConnection/>
                <tag/>
                <url/>
        </scm>
        <properties>
                <java.version>21</java.version>
        </properties>
        <dependencies>
        <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
```

```
   <version>8.3.0</version> <!-- Or latest stable -->
</dependency>
<dependency>
   <groupId>org.springdoc</groupId>
   <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
   <version>2.2.0</version>
</dependency>


                <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.h2database</groupId>
                        <artifactId>h2</artifactId>
                        <scope>runtime</scope>
                </dependency>
                <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>
```