✅ 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. Admin can add, view, and update products.
2. Users can place orders for available products.
3. The system reduces stock when an order is placed.
4. 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<Product, Long>`
- `OrderRepository extends JpaRepository<Order, Long>`

⚙️ 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