Use Case: Clothing Store Management System

Overview

The Clothing Store Management System will allow users to:

- View a list of available clothing items.
- Add new clothing items to the inventory.
- Update existing clothing items.
- Delete clothing items from the inventory.
- Place customer orders for clothing items.
- View order history.

Technology Stack

- **Backend**: Spring Boot
- **Frontend**: Angular
- **Database**: H2 Database (for simplicity, but you can use any relational database)
- **API Communication**: RESTful APIs

```
// src/main/java/com/example/clothingstore/model/ClothingItem.java
package com.example.clothingstore.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class ClothingItem {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
private String name;
private String category;
private double price;
private int stock;

// Getters and Setters
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 String getCategory() {
   return category;
}

public void setCategory(String category) {
   this.category = category;
}

public double getPrice() {
```

```
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public int getStock() {
        return stock;
    }

    public void setStock(int stock) {
        this.stock = stock;
    }
}
```

```
// src/main/java/com/example/clothingstore/model/Order.java
package com.example.clothingstore.model;

import javax.persistence.*;
import java.time.LocalDate;
import java.util.List;

@Entity
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
@ManyToMany
@JoinTable(
    name = "order_clothing_item",
    joinColumns = @JoinColumn(name = "order_id"),
    inverseJoinColumns = @JoinColumn(name = "clothing_item_id")
)
private List<ClothingItem> clothingItems;

private String customerName;
private LocalDate orderDate;

// Getters and Setters
public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public List<ClothingItem> getClothingItems() {
    return clothingItems;
}

public void setClothingItems(List<ClothingItem> clothingItems) {
    this.clothingItems = clothingItems;
}

public String getCustomerName() {
    return customerName;
}
```

```
    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public LocalDate getOrderDate() {
        return orderDate;
    }

    public void setOrderDate(LocalDate orderDate) {
        this.orderDate = orderDate;
    }
}
```

#Implement validation on backed model class

#create jpa repository,service layer and controllers

#Implement global exception handler

#Test backend using post man

#Create front end using angular

#create services in angular to send request through http method

#perform crud operation from front end to backend

#Implement security in both backend and front end