## Coding Task: Order Creation Service

### Core Requirement:

Create a Spring Boot service that allows creating an order with basic validation and returns appropriate responses.

### Domain Models:

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

    @Column(nullable = false)
    private String orderNumber;

    @Column(nullable = false)
    @Enumerated(EnumType.STRING)
    private OrderStatus status;

    @Column(nullable = false)
    private LocalDateTime createdAt;

    @OneToMany(mappedBy = "order", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<OrderItem> items = new ArrayList<>();

    private BigDecimal totalAmount;
}

@Entity
@Table(name = "order_items")
public class OrderItem {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "order_id", nullable = false)
    private Order order;

    @Column(nullable = false)
    private String productCode;

    @Column(nullable = false)
    private Integer quantity;

    @Column(nullable = false)
    private BigDecimal unitPrice;
}
```

```
public enum OrderStatus {
    CREATED,
    VALIDATED,
    FAILED
}
```

**API Endpoint:**

### Create Order API

**POST** `/api/v1/orders`

Request Body:

```
{
    "items": [
        {
            "productCode": "string",
            "quantity": "integer",
            "unitPrice": "decimal"
        }
    ]
}
```

Response: 201 Created

```
{
    "orderNumber": "string",
    "status": "string",
    "totalAmount": "decimal",
    "createdAt": "datetime",
    "items": [
        {
            "productCode": "string",
            "quantity": "integer",
            "unitPrice": "decimal"
        }
    ]
}
```

Error Responses: 400 Bad Request

```
{
    "message": "string",
    "errors": [
        {
            "field": "string",
            "message": "string"
        }
    ]
}
```

**Requirements:**

1. Input Validation:

    - Order must have at least one item
    - Quantity must be positive
    - Unit price must be positive
    - Product code must not be empty

2. Business Logic:

    - Generate unique order number
    - Calculate total amount
    - Set initial status as CREATED
    - Store in database

3. Technical Requirements:

    - Proper exception handling
    - Unit tests
    - Input validation
    - Clean code organization
    - Logging