**Instructions**: In this test, you will be simulating an e-commerce order management system. Please read the scenario and complete the tasks using Python, Django, Django ORM, and Django Rest Framework.

**Scenario**:
You are tasked with building an API for an e-commerce website's order management system. The system should allow users to place orders, view their orders, and update order status. Each order can contain multiple items, and each item has a name, quantity, and price.

**Tasks**:

1. Create a Django model called Order that represents an order. The model should have the following fields:

user (Foreign Key to the User model)
status (choices: "PENDING", "SHIPPED", "DELIVERED")
created_at (DateTimeField)
updated_at (DateTimeField)

2. Create a Django model called OrderItem that represents an item within an order. The model should have the following fields:

   - order (Foreign Key to the Order model)
   - name (CharField)
   - quantity (PositiveIntegerField)
   - price (DecimalField)

3. Create a Django REST Framework serializer for the Order model. The serializer should include all the fields of the Order model and display the nested items within the order.

4. Implement an API endpoint for creating orders. The endpoint should accept a POST request with the following JSON payload:

```
{
  "user": <user_id>,
  "status": "PENDING",
  "items": [
    {"name": "Item 1", "quantity": 2, "price": 10.0},
    {"name": "Item 2", "quantity": 1, "price": 15.0}
  ]
}
```

   The endpoint should create a new order with the provided data.

5. Implement an API endpoint for retrieving a specific order by its ID. The endpoint should accept a GET request with the order ID and return the order details, including the nested items.

6. Implement an API endpoint for updating the status of an order. The endpoint should accept a PUT request with the order ID and the updated status. It should update the order's status field accordingly.

7. Implement an API endpoint for listing all orders. The endpoint should accept a GET request and return a list of all orders with their details, including the nested items.

Note: You can assume that the User model and authentication are already set up in the Django project.

Please complete the tasks and provide the code for the models, serializers, and API views. Feel free to use any additional libraries or packages if necessary.