Django\_api\_views

Here is the Django API CRUD views for OrderItem:  
```python  
# Views for OrderItem model  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.permissions import IsAuthenticated  
from rest\_framework.status import HTTP\_201\_CREATED, HTTP\_204\_NO\_CONTENT, HTTP\_400\_BAD\_REQUEST  
from.models import OrderItem  
from.serializers import OrderItemSerializer  
  
  
class OrderItemList(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get(self, request):  
 order\_items = OrderItem.objects.all()  
 serializer = OrderItemSerializer(order\_items, many=True)  
 return Response(serializer.data)  
  
 def post(self, request):  
 serializer = OrderItemSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
  
class OrderItemDetail(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get\_object(self, pk):  
 try:  
 return OrderItem.objects.get(pk=pk)  
 except OrderItem.DoesNotExist:  
 return Response(status=HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item)  
 return Response(serializer.data)  
  
 def put(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
 def delete(self, request, pk):  
 order\_item = self.get\_object(pk)  
 order\_item.delete()  
 return Response(status=HTTP\_204\_NO\_CONTENT)  
```  
  
In these views, we have two classes: `OrderItemList` and `OrderItemDetail`. The `OrderItemList` view handles GET and POST requests for all OrderItems. The `OrderItemDetail` view handles GET, PUT, and DELETE requests for a single OrderItem identified by its primary key (`pk`). These views use the `OrderItemSerializer` to serialize and deserialize OrderItem objects into JSON. They also implement basic authentication using the `IsAuthenticated` permission class from Django Rest Framework.```  
  
  
  
```python  
# serializers.py  
from rest\_framework import serializers  
from.models import OrderItem  
  
class OrderItemSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = OrderItem  
 fields = ['id', 'order', 'product', 'quantity','subtotal']  
```  
  
I hope you find it helpful! If you need further assistance or clarification, feel free to ask.

Here is the Django API CRUD views for OrderItem:  
```python  
# Views for OrderItem model  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.permissions import IsAuthenticated  
from rest\_framework.status import HTTP\_201\_CREATED, HTTP\_204\_NO\_CONTENT, HTTP\_400\_BAD\_REQUEST  
from.models import OrderItem  
from.serializers import OrderItemSerializer  
  
  
class OrderItemList(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get(self, request):  
 order\_items = OrderItem.objects.all()  
 serializer = OrderItemSerializer(order\_items, many=True)  
 return Response(serializer.data)  
  
 def post(self, request):  
 serializer = OrderItemSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
  
class OrderItemDetail(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get\_object(self, pk):  
 try:  
 return OrderItem.objects.get(pk=pk)  
 except OrderItem.DoesNotExist:  
 return Response(status=HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item)  
 return Response(serializer.data)  
  
 def put(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
 def delete(self, request, pk):  
 order\_item = self.get\_object(pk)  
 order\_item.delete()  
 return Response(status=HTTP\_204\_NO\_CONTENT)  
```  
  
In these views, we have two classes: `OrderItemList` and `OrderItemDetail`. The `OrderItemList` view handles GET and POST requests for all OrderItems. The `OrderItemDetail` view handles GET, PUT, and DELETE requests for a single OrderItem identified by its primary key (`pk`). These views use the `OrderItemSerializer` to serialize and deserialize OrderItem objects into JSON. They also implement basic authentication using the `IsAuthenticated` permission class from Django Rest Framework.```  
  
  
  
```python  
# serializers.py  
from rest\_framework import serializers  
from.models import OrderItem  
  
class OrderItemSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = OrderItem  
 fields = ['id', 'order', 'product', 'quantity','subtotal']  
```  
  
I hope you find it helpful! If you need further assistance or clarification, feel free to ask.

Here is the Django API CRUD views for OrderItem:  
```python  
# Views for OrderItem model  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.permissions import IsAuthenticated  
from rest\_framework.status import HTTP\_201\_CREATED, HTTP\_204\_NO\_CONTENT, HTTP\_400\_BAD\_REQUEST  
from.models import OrderItem  
from.serializers import OrderItemSerializer  
  
  
class OrderItemList(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get(self, request):  
 order\_items = OrderItem.objects.all()  
 serializer = OrderItemSerializer(order\_items, many=True)  
 return Response(serializer.data)  
  
 def post(self, request):  
 serializer = OrderItemSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
  
class OrderItemDetail(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get\_object(self, pk):  
 try:  
 return OrderItem.objects.get(pk=pk)  
 except OrderItem.DoesNotExist:  
 return Response(status=HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item)  
 return Response(serializer.data)  
  
 def put(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
 def delete(self, request, pk):  
 order\_item = self.get\_object(pk)  
 order\_item.delete()  
 return Response(status=HTTP\_204\_NO\_CONTENT)  
```  
  
In these views, we have two classes: `OrderItemList` and `OrderItemDetail`. The `OrderItemList` view handles GET and POST requests for all OrderItems. The `OrderItemDetail` view handles GET, PUT, and DELETE requests for a single OrderItem identified by its primary key (`pk`). These views use the `OrderItemSerializer` to serialize and deserialize OrderItem objects into JSON. They also implement basic authentication using the `IsAuthenticated` permission class from Django Rest Framework.```  
  
  
  
```python  
# serializers.py  
from rest\_framework import serializers  
from.models import OrderItem  
  
class OrderItemSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = OrderItem  
 fields = ['id', 'order', 'product', 'quantity','subtotal']  
```  
  
I hope you find it helpful! If you need further assistance or clarification, feel free to ask.

Here is the Django API CRUD views for OrderItem:  
```python  
# Views for OrderItem model  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.permissions import IsAuthenticated  
from rest\_framework.status import HTTP\_201\_CREATED, HTTP\_204\_NO\_CONTENT, HTTP\_400\_BAD\_REQUEST  
from.models import OrderItem  
from.serializers import OrderItemSerializer  
  
  
class OrderItemList(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get(self, request):  
 order\_items = OrderItem.objects.all()  
 serializer = OrderItemSerializer(order\_items, many=True)  
 return Response(serializer.data)  
  
 def post(self, request):  
 serializer = OrderItemSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
  
class OrderItemDetail(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get\_object(self, pk):  
 try:  
 return OrderItem.objects.get(pk=pk)  
 except OrderItem.DoesNotExist:  
 return Response(status=HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item)  
 return Response(serializer.data)  
  
 def put(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
 def delete(self, request, pk):  
 order\_item = self.get\_object(pk)  
 order\_item.delete()  
 return Response(status=HTTP\_204\_NO\_CONTENT)  
```  
  
In these views, we have two classes: `OrderItemList` and `OrderItemDetail`. The `OrderItemList` view handles GET and POST requests for all OrderItems. The `OrderItemDetail` view handles GET, PUT, and DELETE requests for a single OrderItem identified by its primary key (`pk`). These views use the `OrderItemSerializer` to serialize and deserialize OrderItem objects into JSON. They also implement basic authentication using the `IsAuthenticated` permission class from Django Rest Framework.```  
  
  
  
```python  
# serializers.py  
from rest\_framework import serializers  
from.models import OrderItem  
  
class OrderItemSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = OrderItem  
 fields = ['id', 'order', 'product', 'quantity','subtotal']  
```  
  
I hope you find it helpful! If you need further assistance or clarification, feel free to ask.

Here is the Django API CRUD views for OrderItem:  
```python  
# Views for OrderItem model  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.permissions import IsAuthenticated  
from rest\_framework.status import HTTP\_201\_CREATED, HTTP\_204\_NO\_CONTENT, HTTP\_400\_BAD\_REQUEST  
from.models import OrderItem  
from.serializers import OrderItemSerializer  
  
  
class OrderItemList(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get(self, request):  
 order\_items = OrderItem.objects.all()  
 serializer = OrderItemSerializer(order\_items, many=True)  
 return Response(serializer.data)  
  
 def post(self, request):  
 serializer = OrderItemSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
  
class OrderItemDetail(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get\_object(self, pk):  
 try:  
 return OrderItem.objects.get(pk=pk)  
 except OrderItem.DoesNotExist:  
 return Response(status=HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item)  
 return Response(serializer.data)  
  
 def put(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
 def delete(self, request, pk):  
 order\_item = self.get\_object(pk)  
 order\_item.delete()  
 return Response(status=HTTP\_204\_NO\_CONTENT)  
```  
  
In these views, we have two classes: `OrderItemList` and `OrderItemDetail`. The `OrderItemList` view handles GET and POST requests for all OrderItems. The `OrderItemDetail` view handles GET, PUT, and DELETE requests for a single OrderItem identified by its primary key (`pk`). These views use the `OrderItemSerializer` to serialize and deserialize OrderItem objects into JSON. They also implement basic authentication using the `IsAuthenticated` permission class from Django Rest Framework.```  
  
  
  
```python  
# serializers.py  
from rest\_framework import serializers  
from.models import OrderItem  
  
class OrderItemSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = OrderItem  
 fields = ['id', 'order', 'product', 'quantity','subtotal']  
```  
  
I hope you find it helpful! If you need further assistance or clarification, feel free to ask.

Here is the Django API CRUD views for OrderItem:  
```python  
# Views for OrderItem model  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.permissions import IsAuthenticated  
from rest\_framework.status import HTTP\_201\_CREATED, HTTP\_204\_NO\_CONTENT, HTTP\_400\_BAD\_REQUEST  
from.models import OrderItem  
from.serializers import OrderItemSerializer  
  
  
class OrderItemList(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get(self, request):  
 order\_items = OrderItem.objects.all()  
 serializer = OrderItemSerializer(order\_items, many=True)  
 return Response(serializer.data)  
  
 def post(self, request):  
 serializer = OrderItemSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
  
class OrderItemDetail(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get\_object(self, pk):  
 try:  
 return OrderItem.objects.get(pk=pk)  
 except OrderItem.DoesNotExist:  
 return Response(status=HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item)  
 return Response(serializer.data)  
  
 def put(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
 def delete(self, request, pk):  
 order\_item = self.get\_object(pk)  
 order\_item.delete()  
 return Response(status=HTTP\_204\_NO\_CONTENT)  
```  
  
In these views, we have two classes: `OrderItemList` and `OrderItemDetail`. The `OrderItemList` view handles GET and POST requests for all OrderItems. The `OrderItemDetail` view handles GET, PUT, and DELETE requests for a single OrderItem identified by its primary key (`pk`). These views use the `OrderItemSerializer` to serialize and deserialize OrderItem objects into JSON. They also implement basic authentication using the `IsAuthenticated` permission class from Django Rest Framework.```  
  
  
  
```python  
# serializers.py  
from rest\_framework import serializers  
from.models import OrderItem  
  
class OrderItemSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = OrderItem  
 fields = ['id', 'order', 'product', 'quantity','subtotal']  
```  
  
I hope you find it helpful! If you need further assistance or clarification, feel free to ask.

Here is the Django API CRUD views for OrderItem:  
```python  
# Views for OrderItem model  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.permissions import IsAuthenticated  
from rest\_framework.status import HTTP\_201\_CREATED, HTTP\_204\_NO\_CONTENT, HTTP\_400\_BAD\_REQUEST  
from.models import OrderItem  
from.serializers import OrderItemSerializer  
  
  
class OrderItemList(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get(self, request):  
 order\_items = OrderItem.objects.all()  
 serializer = OrderItemSerializer(order\_items, many=True)  
 return Response(serializer.data)  
  
 def post(self, request):  
 serializer = OrderItemSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
  
class OrderItemDetail(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get\_object(self, pk):  
 try:  
 return OrderItem.objects.get(pk=pk)  
 except OrderItem.DoesNotExist:  
 return Response(status=HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item)  
 return Response(serializer.data)  
  
 def put(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
 def delete(self, request, pk):  
 order\_item = self.get\_object(pk)  
 order\_item.delete()  
 return Response(status=HTTP\_204\_NO\_CONTENT)  
```  
  
In these views, we have two classes: `OrderItemList` and `OrderItemDetail`. The `OrderItemList` view handles GET and POST requests for all OrderItems. The `OrderItemDetail` view handles GET, PUT, and DELETE requests for a single OrderItem identified by its primary key (`pk`). These views use the `OrderItemSerializer` to serialize and deserialize OrderItem objects into JSON. They also implement basic authentication using the `IsAuthenticated` permission class from Django Rest Framework.```  
  
  
  
```python  
# serializers.py  
from rest\_framework import serializers  
from.models import OrderItem  
  
class OrderItemSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = OrderItem  
 fields = ['id', 'order', 'product', 'quantity','subtotal']  
```  
  
I hope you find it helpful! If you need further assistance or clarification, feel free to ask.

Here is the Django API CRUD views for OrderItem:  
```python  
# Views for OrderItem model  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.permissions import IsAuthenticated  
from rest\_framework.status import HTTP\_201\_CREATED, HTTP\_204\_NO\_CONTENT, HTTP\_400\_BAD\_REQUEST  
from.models import OrderItem  
from.serializers import OrderItemSerializer  
  
  
class OrderItemList(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get(self, request):  
 order\_items = OrderItem.objects.all()  
 serializer = OrderItemSerializer(order\_items, many=True)  
 return Response(serializer.data)  
  
 def post(self, request):  
 serializer = OrderItemSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
  
class OrderItemDetail(APIView):  
 permission\_classes = [IsAuthenticated]  
  
 def get\_object(self, pk):  
 try:  
 return OrderItem.objects.get(pk=pk)  
 except OrderItem.DoesNotExist:  
 return Response(status=HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item)  
 return Response(serializer.data)  
  
 def put(self, request, pk):  
 order\_item = self.get\_object(pk)  
 serializer = OrderItemSerializer(order\_item, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=HTTP\_400\_BAD\_REQUEST)  
  
 def delete(self, request, pk):  
 order\_item = self.get\_object(pk)  
 order\_item.delete()  
 return Response(status=HTTP\_204\_NO\_CONTENT)  
```  
  
In these views, we have two classes: `OrderItemList` and `OrderItemDetail`. The `OrderItemList` view handles GET and POST requests for all OrderItems. The `OrderItemDetail` view handles GET, PUT, and DELETE requests for a single OrderItem identified by its primary key (`pk`). These views use the `OrderItemSerializer` to serialize and deserialize OrderItem objects into JSON. They also implement basic authentication using the `IsAuthenticated` permission class from Django Rest Framework.```  
  
  
  
```python  
# serializers.py  
from rest\_framework import serializers  
from.models import OrderItem  
  
class OrderItemSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = OrderItem  
 fields = ['id', 'order', 'product', 'quantity','subtotal']  
```  
  
I hope you find it helpful! If you need further assistance or clarification, feel free to ask.