Django\_serializer

Here is the Django serializer code for the Review model:  
```python  
from rest\_framework import serializers  
from.models import Review  
  
class ReviewSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Review  
 fields = ['review\_id', 'customer\_id', 'product\_id', 'rating', 'comment']  
```  
  
In this code, I have created a serializer class called `ReviewSerializer` that inherits from `serializers.ModelSerializer`. This serializer is used to convert complex data such as QuerySets and model instances into native Python datatypes that can then be easily rendered into JSON or XML format. In this case, we are mapping the fields of the `Review` model to specific fields in our API response.   
  
Please let me know if you need anything else!

Here is the Django serializer code for the Review model:  
```python  
from rest\_framework import serializers  
from.models import Review  
  
class ReviewSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Review  
 fields = ['review\_id', 'customer\_id', 'product\_id', 'rating', 'comment']  
```  
  
In this code, I have created a serializer class called `ReviewSerializer` that inherits from `serializers.ModelSerializer`. This serializer is used to convert complex data such as QuerySets and model instances into native Python datatypes that can then be easily rendered into JSON or XML format. In this case, we are mapping the fields of the `Review` model to specific fields in our API response.   
  
Please let me know if you need anything else!

Here is the Django serializer code for the Review model:  
```python  
from rest\_framework import serializers  
from.models import Review  
  
class ReviewSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Review  
 fields = ['review\_id', 'customer\_id', 'product\_id', 'rating', 'comment']  
```  
  
In this code, I have created a serializer class called `ReviewSerializer` that inherits from `serializers.ModelSerializer`. This serializer is used to convert complex data such as QuerySets and model instances into native Python datatypes that can then be easily rendered into JSON or XML format. In this case, we are mapping the fields of the `Review` model to specific fields in our API response.   
  
Please let me know if you need anything else!

Here is the Django serializer code for the Review model:  
```python  
from rest\_framework import serializers  
from.models import Review  
  
class ReviewSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Review  
 fields = ['review\_id', 'customer\_id', 'product\_id', 'rating', 'comment']  
```  
  
In this code, I have created a serializer class called `ReviewSerializer` that inherits from `serializers.ModelSerializer`. This serializer is used to convert complex data such as QuerySets and model instances into native Python datatypes that can then be easily rendered into JSON or XML format. In this case, we are mapping the fields of the `Review` model to specific fields in our API response.   
  
Please let me know if you need anything else!

Here is the Django serializer code for the Review model:  
```python  
from rest\_framework import serializers  
from.models import Review  
  
class ReviewSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Review  
 fields = ['review\_id', 'customer\_id', 'product\_id', 'rating', 'comment']  
```  
  
In this code, I have created a serializer class called `ReviewSerializer` that inherits from `serializers.ModelSerializer`. This serializer is used to convert complex data such as QuerySets and model instances into native Python datatypes that can then be easily rendered into JSON or XML format. In this case, we are mapping the fields of the `Review` model to specific fields in our API response.   
  
Please let me know if you need anything else!

Here is the Django serializer code for the Review model:  
```python  
from rest\_framework import serializers  
from.models import Review  
  
class ReviewSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Review  
 fields = ['review\_id', 'customer\_id', 'product\_id', 'rating', 'comment']  
```  
  
In this code, I have created a serializer class called `ReviewSerializer` that inherits from `serializers.ModelSerializer`. This serializer is used to convert complex data such as QuerySets and model instances into native Python datatypes that can then be easily rendered into JSON or XML format. In this case, we are mapping the fields of the `Review` model to specific fields in our API response.   
  
Please let me know if you need anything else!

Here is the Django serializer code for the Review model:  
```python  
from rest\_framework import serializers  
from.models import Review  
  
class ReviewSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Review  
 fields = ['review\_id', 'customer\_id', 'product\_id', 'rating', 'comment']  
```  
  
In this code, I have created a serializer class called `ReviewSerializer` that inherits from `serializers.ModelSerializer`. This serializer is used to convert complex data such as QuerySets and model instances into native Python datatypes that can then be easily rendered into JSON or XML format. In this case, we are mapping the fields of the `Review` model to specific fields in our API response.   
  
Please let me know if you need anything else!

Here is the Django serializer code for the Review model:  
```python  
from rest\_framework import serializers  
from.models import Review  
  
class ReviewSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Review  
 fields = ['review\_id', 'customer\_id', 'product\_id', 'rating', 'comment']  
```  
  
In this code, I have created a serializer class called `ReviewSerializer` that inherits from `serializers.ModelSerializer`. This serializer is used to convert complex data such as QuerySets and model instances into native Python datatypes that can then be easily rendered into JSON or XML format. In this case, we are mapping the fields of the `Review` model to specific fields in our API response.   
  
Please let me know if you need anything else!