Django\_api\_views

Here's my response in Python/Django syntax:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework import status  
from.models import Review  
from.serializers import ReviewSerializer  
  
# Create your views here.  
class ReviewList(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET all reviews  
 def get(self, request):  
 review\_list = self.queryset.all()  
 serializer = self.serializer\_class(review\_list, many=True)  
 return Response(serializer.data)  
  
 # POST new review  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
  
class ReviewDetail(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET single review  
 def get\_object(self, pk):  
 try:  
 return Review.objects.get(pk=pk)  
 except Review.DoesNotExist:  
 return Response(status=status.HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review)  
 return Response(serializer.data)  
  
 # PUT update review  
 def put(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
 # DELETE review  
 def delete(self, request, pk):  
 review = self.get\_object(pk)  
 review.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
  
This code provides the necessary CRUD operations for the Reviews model using Django Rest Framework's APIView class. The ReviewList view handles listing and creating reviews, while the ReviewDetail view handles retrieving, updating, and deleting individual reviews.  
  
I hope this helps! Let me know if you have any questions or need further clarification.

Here's my response in Python/Django syntax:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework import status  
from.models import Review  
from.serializers import ReviewSerializer  
  
# Create your views here.  
class ReviewList(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET all reviews  
 def get(self, request):  
 review\_list = self.queryset.all()  
 serializer = self.serializer\_class(review\_list, many=True)  
 return Response(serializer.data)  
  
 # POST new review  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
  
class ReviewDetail(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET single review  
 def get\_object(self, pk):  
 try:  
 return Review.objects.get(pk=pk)  
 except Review.DoesNotExist:  
 return Response(status=status.HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review)  
 return Response(serializer.data)  
  
 # PUT update review  
 def put(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
 # DELETE review  
 def delete(self, request, pk):  
 review = self.get\_object(pk)  
 review.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
  
This code provides the necessary CRUD operations for the Reviews model using Django Rest Framework's APIView class. The ReviewList view handles listing and creating reviews, while the ReviewDetail view handles retrieving, updating, and deleting individual reviews.  
  
I hope this helps! Let me know if you have any questions or need further clarification.

Here's my response in Python/Django syntax:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework import status  
from.models import Review  
from.serializers import ReviewSerializer  
  
# Create your views here.  
class ReviewList(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET all reviews  
 def get(self, request):  
 review\_list = self.queryset.all()  
 serializer = self.serializer\_class(review\_list, many=True)  
 return Response(serializer.data)  
  
 # POST new review  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
  
class ReviewDetail(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET single review  
 def get\_object(self, pk):  
 try:  
 return Review.objects.get(pk=pk)  
 except Review.DoesNotExist:  
 return Response(status=status.HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review)  
 return Response(serializer.data)  
  
 # PUT update review  
 def put(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
 # DELETE review  
 def delete(self, request, pk):  
 review = self.get\_object(pk)  
 review.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
  
This code provides the necessary CRUD operations for the Reviews model using Django Rest Framework's APIView class. The ReviewList view handles listing and creating reviews, while the ReviewDetail view handles retrieving, updating, and deleting individual reviews.  
  
I hope this helps! Let me know if you have any questions or need further clarification.

Here's my response in Python/Django syntax:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework import status  
from.models import Review  
from.serializers import ReviewSerializer  
  
# Create your views here.  
class ReviewList(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET all reviews  
 def get(self, request):  
 review\_list = self.queryset.all()  
 serializer = self.serializer\_class(review\_list, many=True)  
 return Response(serializer.data)  
  
 # POST new review  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
  
class ReviewDetail(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET single review  
 def get\_object(self, pk):  
 try:  
 return Review.objects.get(pk=pk)  
 except Review.DoesNotExist:  
 return Response(status=status.HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review)  
 return Response(serializer.data)  
  
 # PUT update review  
 def put(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
 # DELETE review  
 def delete(self, request, pk):  
 review = self.get\_object(pk)  
 review.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
  
This code provides the necessary CRUD operations for the Reviews model using Django Rest Framework's APIView class. The ReviewList view handles listing and creating reviews, while the ReviewDetail view handles retrieving, updating, and deleting individual reviews.  
  
I hope this helps! Let me know if you have any questions or need further clarification.

Here's my response in Python/Django syntax:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework import status  
from.models import Review  
from.serializers import ReviewSerializer  
  
# Create your views here.  
class ReviewList(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET all reviews  
 def get(self, request):  
 review\_list = self.queryset.all()  
 serializer = self.serializer\_class(review\_list, many=True)  
 return Response(serializer.data)  
  
 # POST new review  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
  
class ReviewDetail(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET single review  
 def get\_object(self, pk):  
 try:  
 return Review.objects.get(pk=pk)  
 except Review.DoesNotExist:  
 return Response(status=status.HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review)  
 return Response(serializer.data)  
  
 # PUT update review  
 def put(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
 # DELETE review  
 def delete(self, request, pk):  
 review = self.get\_object(pk)  
 review.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
  
This code provides the necessary CRUD operations for the Reviews model using Django Rest Framework's APIView class. The ReviewList view handles listing and creating reviews, while the ReviewDetail view handles retrieving, updating, and deleting individual reviews.  
  
I hope this helps! Let me know if you have any questions or need further clarification.

Here's my response in Python/Django syntax:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework import status  
from.models import Review  
from.serializers import ReviewSerializer  
  
# Create your views here.  
class ReviewList(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET all reviews  
 def get(self, request):  
 review\_list = self.queryset.all()  
 serializer = self.serializer\_class(review\_list, many=True)  
 return Response(serializer.data)  
  
 # POST new review  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
  
class ReviewDetail(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET single review  
 def get\_object(self, pk):  
 try:  
 return Review.objects.get(pk=pk)  
 except Review.DoesNotExist:  
 return Response(status=status.HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review)  
 return Response(serializer.data)  
  
 # PUT update review  
 def put(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
 # DELETE review  
 def delete(self, request, pk):  
 review = self.get\_object(pk)  
 review.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
  
This code provides the necessary CRUD operations for the Reviews model using Django Rest Framework's APIView class. The ReviewList view handles listing and creating reviews, while the ReviewDetail view handles retrieving, updating, and deleting individual reviews.  
  
I hope this helps! Let me know if you have any questions or need further clarification.

Here's my response in Python/Django syntax:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework import status  
from.models import Review  
from.serializers import ReviewSerializer  
  
# Create your views here.  
class ReviewList(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET all reviews  
 def get(self, request):  
 review\_list = self.queryset.all()  
 serializer = self.serializer\_class(review\_list, many=True)  
 return Response(serializer.data)  
  
 # POST new review  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
  
class ReviewDetail(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET single review  
 def get\_object(self, pk):  
 try:  
 return Review.objects.get(pk=pk)  
 except Review.DoesNotExist:  
 return Response(status=status.HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review)  
 return Response(serializer.data)  
  
 # PUT update review  
 def put(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
 # DELETE review  
 def delete(self, request, pk):  
 review = self.get\_object(pk)  
 review.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
  
This code provides the necessary CRUD operations for the Reviews model using Django Rest Framework's APIView class. The ReviewList view handles listing and creating reviews, while the ReviewDetail view handles retrieving, updating, and deleting individual reviews.  
  
I hope this helps! Let me know if you have any questions or need further clarification.

Here's my response in Python/Django syntax:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework import status  
from.models import Review  
from.serializers import ReviewSerializer  
  
# Create your views here.  
class ReviewList(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET all reviews  
 def get(self, request):  
 review\_list = self.queryset.all()  
 serializer = self.serializer\_class(review\_list, many=True)  
 return Response(serializer.data)  
  
 # POST new review  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
  
class ReviewDetail(APIView):  
 queryset = Review.objects.all()  
 serializer\_class = ReviewSerializer  
  
 # GET single review  
 def get\_object(self, pk):  
 try:  
 return Review.objects.get(pk=pk)  
 except Review.DoesNotExist:  
 return Response(status=status.HTTP\_404\_NOT\_FOUND)  
  
 def get(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review)  
 return Response(serializer.data)  
  
 # PUT update review  
 def put(self, request, pk):  
 review = self.get\_object(pk)  
 serializer = self.serializer\_class(review, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
 # DELETE review  
 def delete(self, request, pk):  
 review = self.get\_object(pk)  
 review.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
  
This code provides the necessary CRUD operations for the Reviews model using Django Rest Framework's APIView class. The ReviewList view handles listing and creating reviews, while the ReviewDetail view handles retrieving, updating, and deleting individual reviews.  
  
I hope this helps! Let me know if you have any questions or need further clarification.