Django\_views

Here is the views.py file for CRUD operations on the Reviews Model:  
  
```python  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.parsers import JSONParser  
from rest\_framework.renderers import JSONRenderer  
from.models import Review  
from.serializers import ReviewSerializer  
import json  
# Create your views here.  
  
  
class ReviewList(APIView):  
  
 # GET method to get all reviews  
 def get(self,request):  
 review=Review.objects.all()  
 serializer=ReviewSerializer(review,many=True)  
 return Response(serializer.data)  
  
  
 # POST method to create new review  
 def post(self,request):  
 data=request.data  
 serializer=ReviewSerializer(data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'review added'})  
 else:  
 return Response(serializer.errors)  
  
  
class ReviewDetail(APIView):  
  
 # GET method to retrieve specific review details  
 def get\_object(self,id):  
 try:  
 return Review.objects.get(id=id)  
 except Exception as e:  
 return None  
  
 def get(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 serializer=ReviewSerializer(review)  
 return Response(serializer.data)  
 else:  
 return Response({'msg':'no record found'})  
  
 # PUT method to update existing review  
 def put(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 data=request.data  
 serializer=ReviewSerializer(review,data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'updated successfully'})  
 else:  
 return Response(serializer.errors)  
 else:  
 return Response({'msg':'no record found'})  
  
  
  
 # DELETE method to delete review  
 def delete(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 review.delete()  
 return Response({'msg':'deleted successfully'})  
 else:  
 return Response({'msg':'no record found'})  
```  
  
This view handles the following actions:  
1. \*\*GET /reviews\*\*: Returns a list of all reviews.  
2. \*\*POST /reviews\*\*: Creates a new review.  
3. \*\*GET /reviews/{id}\*\*: Retrieves the details of a specific review.  
4. \*\*PUT /reviews/{id}\*\*: Updates an existing review.  
5. \*\*DELETE /reviews/{id}\*\*: Deletes a review.  
  
Note that this is just one way to implement these views, there might be other ways depending on how you want to structure your code.  
  
I hope it helps! Let me know if you have any questions or need further assistance.

Here is the views.py file for CRUD operations on the Reviews Model:  
  
```python  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.parsers import JSONParser  
from rest\_framework.renderers import JSONRenderer  
from.models import Review  
from.serializers import ReviewSerializer  
import json  
# Create your views here.  
  
  
class ReviewList(APIView):  
  
 # GET method to get all reviews  
 def get(self,request):  
 review=Review.objects.all()  
 serializer=ReviewSerializer(review,many=True)  
 return Response(serializer.data)  
  
  
 # POST method to create new review  
 def post(self,request):  
 data=request.data  
 serializer=ReviewSerializer(data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'review added'})  
 else:  
 return Response(serializer.errors)  
  
  
class ReviewDetail(APIView):  
  
 # GET method to retrieve specific review details  
 def get\_object(self,id):  
 try:  
 return Review.objects.get(id=id)  
 except Exception as e:  
 return None  
  
 def get(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 serializer=ReviewSerializer(review)  
 return Response(serializer.data)  
 else:  
 return Response({'msg':'no record found'})  
  
 # PUT method to update existing review  
 def put(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 data=request.data  
 serializer=ReviewSerializer(review,data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'updated successfully'})  
 else:  
 return Response(serializer.errors)  
 else:  
 return Response({'msg':'no record found'})  
  
  
  
 # DELETE method to delete review  
 def delete(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 review.delete()  
 return Response({'msg':'deleted successfully'})  
 else:  
 return Response({'msg':'no record found'})  
```  
  
This view handles the following actions:  
1. \*\*GET /reviews\*\*: Returns a list of all reviews.  
2. \*\*POST /reviews\*\*: Creates a new review.  
3. \*\*GET /reviews/{id}\*\*: Retrieves the details of a specific review.  
4. \*\*PUT /reviews/{id}\*\*: Updates an existing review.  
5. \*\*DELETE /reviews/{id}\*\*: Deletes a review.  
  
Note that this is just one way to implement these views, there might be other ways depending on how you want to structure your code.  
  
I hope it helps! Let me know if you have any questions or need further assistance.

Here is the views.py file for CRUD operations on the Reviews Model:  
  
```python  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.parsers import JSONParser  
from rest\_framework.renderers import JSONRenderer  
from.models import Review  
from.serializers import ReviewSerializer  
import json  
# Create your views here.  
  
  
class ReviewList(APIView):  
  
 # GET method to get all reviews  
 def get(self,request):  
 review=Review.objects.all()  
 serializer=ReviewSerializer(review,many=True)  
 return Response(serializer.data)  
  
  
 # POST method to create new review  
 def post(self,request):  
 data=request.data  
 serializer=ReviewSerializer(data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'review added'})  
 else:  
 return Response(serializer.errors)  
  
  
class ReviewDetail(APIView):  
  
 # GET method to retrieve specific review details  
 def get\_object(self,id):  
 try:  
 return Review.objects.get(id=id)  
 except Exception as e:  
 return None  
  
 def get(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 serializer=ReviewSerializer(review)  
 return Response(serializer.data)  
 else:  
 return Response({'msg':'no record found'})  
  
 # PUT method to update existing review  
 def put(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 data=request.data  
 serializer=ReviewSerializer(review,data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'updated successfully'})  
 else:  
 return Response(serializer.errors)  
 else:  
 return Response({'msg':'no record found'})  
  
  
  
 # DELETE method to delete review  
 def delete(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 review.delete()  
 return Response({'msg':'deleted successfully'})  
 else:  
 return Response({'msg':'no record found'})  
```  
  
This view handles the following actions:  
1. \*\*GET /reviews\*\*: Returns a list of all reviews.  
2. \*\*POST /reviews\*\*: Creates a new review.  
3. \*\*GET /reviews/{id}\*\*: Retrieves the details of a specific review.  
4. \*\*PUT /reviews/{id}\*\*: Updates an existing review.  
5. \*\*DELETE /reviews/{id}\*\*: Deletes a review.  
  
Note that this is just one way to implement these views, there might be other ways depending on how you want to structure your code.  
  
I hope it helps! Let me know if you have any questions or need further assistance.

Here is the views.py file for CRUD operations on the Reviews Model:  
  
```python  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.parsers import JSONParser  
from rest\_framework.renderers import JSONRenderer  
from.models import Review  
from.serializers import ReviewSerializer  
import json  
# Create your views here.  
  
  
class ReviewList(APIView):  
  
 # GET method to get all reviews  
 def get(self,request):  
 review=Review.objects.all()  
 serializer=ReviewSerializer(review,many=True)  
 return Response(serializer.data)  
  
  
 # POST method to create new review  
 def post(self,request):  
 data=request.data  
 serializer=ReviewSerializer(data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'review added'})  
 else:  
 return Response(serializer.errors)  
  
  
class ReviewDetail(APIView):  
  
 # GET method to retrieve specific review details  
 def get\_object(self,id):  
 try:  
 return Review.objects.get(id=id)  
 except Exception as e:  
 return None  
  
 def get(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 serializer=ReviewSerializer(review)  
 return Response(serializer.data)  
 else:  
 return Response({'msg':'no record found'})  
  
 # PUT method to update existing review  
 def put(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 data=request.data  
 serializer=ReviewSerializer(review,data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'updated successfully'})  
 else:  
 return Response(serializer.errors)  
 else:  
 return Response({'msg':'no record found'})  
  
  
  
 # DELETE method to delete review  
 def delete(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 review.delete()  
 return Response({'msg':'deleted successfully'})  
 else:  
 return Response({'msg':'no record found'})  
```  
  
This view handles the following actions:  
1. \*\*GET /reviews\*\*: Returns a list of all reviews.  
2. \*\*POST /reviews\*\*: Creates a new review.  
3. \*\*GET /reviews/{id}\*\*: Retrieves the details of a specific review.  
4. \*\*PUT /reviews/{id}\*\*: Updates an existing review.  
5. \*\*DELETE /reviews/{id}\*\*: Deletes a review.  
  
Note that this is just one way to implement these views, there might be other ways depending on how you want to structure your code.  
  
I hope it helps! Let me know if you have any questions or need further assistance.

Here is the views.py file for CRUD operations on the Reviews Model:  
  
```python  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.parsers import JSONParser  
from rest\_framework.renderers import JSONRenderer  
from.models import Review  
from.serializers import ReviewSerializer  
import json  
# Create your views here.  
  
  
class ReviewList(APIView):  
  
 # GET method to get all reviews  
 def get(self,request):  
 review=Review.objects.all()  
 serializer=ReviewSerializer(review,many=True)  
 return Response(serializer.data)  
  
  
 # POST method to create new review  
 def post(self,request):  
 data=request.data  
 serializer=ReviewSerializer(data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'review added'})  
 else:  
 return Response(serializer.errors)  
  
  
class ReviewDetail(APIView):  
  
 # GET method to retrieve specific review details  
 def get\_object(self,id):  
 try:  
 return Review.objects.get(id=id)  
 except Exception as e:  
 return None  
  
 def get(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 serializer=ReviewSerializer(review)  
 return Response(serializer.data)  
 else:  
 return Response({'msg':'no record found'})  
  
 # PUT method to update existing review  
 def put(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 data=request.data  
 serializer=ReviewSerializer(review,data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'updated successfully'})  
 else:  
 return Response(serializer.errors)  
 else:  
 return Response({'msg':'no record found'})  
  
  
  
 # DELETE method to delete review  
 def delete(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 review.delete()  
 return Response({'msg':'deleted successfully'})  
 else:  
 return Response({'msg':'no record found'})  
```  
  
This view handles the following actions:  
1. \*\*GET /reviews\*\*: Returns a list of all reviews.  
2. \*\*POST /reviews\*\*: Creates a new review.  
3. \*\*GET /reviews/{id}\*\*: Retrieves the details of a specific review.  
4. \*\*PUT /reviews/{id}\*\*: Updates an existing review.  
5. \*\*DELETE /reviews/{id}\*\*: Deletes a review.  
  
Note that this is just one way to implement these views, there might be other ways depending on how you want to structure your code.  
  
I hope it helps! Let me know if you have any questions or need further assistance.

Here is the views.py file for CRUD operations on the Reviews Model:  
  
```python  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.parsers import JSONParser  
from rest\_framework.renderers import JSONRenderer  
from.models import Review  
from.serializers import ReviewSerializer  
import json  
# Create your views here.  
  
  
class ReviewList(APIView):  
  
 # GET method to get all reviews  
 def get(self,request):  
 review=Review.objects.all()  
 serializer=ReviewSerializer(review,many=True)  
 return Response(serializer.data)  
  
  
 # POST method to create new review  
 def post(self,request):  
 data=request.data  
 serializer=ReviewSerializer(data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'review added'})  
 else:  
 return Response(serializer.errors)  
  
  
class ReviewDetail(APIView):  
  
 # GET method to retrieve specific review details  
 def get\_object(self,id):  
 try:  
 return Review.objects.get(id=id)  
 except Exception as e:  
 return None  
  
 def get(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 serializer=ReviewSerializer(review)  
 return Response(serializer.data)  
 else:  
 return Response({'msg':'no record found'})  
  
 # PUT method to update existing review  
 def put(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 data=request.data  
 serializer=ReviewSerializer(review,data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'updated successfully'})  
 else:  
 return Response(serializer.errors)  
 else:  
 return Response({'msg':'no record found'})  
  
  
  
 # DELETE method to delete review  
 def delete(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 review.delete()  
 return Response({'msg':'deleted successfully'})  
 else:  
 return Response({'msg':'no record found'})  
```  
  
This view handles the following actions:  
1. \*\*GET /reviews\*\*: Returns a list of all reviews.  
2. \*\*POST /reviews\*\*: Creates a new review.  
3. \*\*GET /reviews/{id}\*\*: Retrieves the details of a specific review.  
4. \*\*PUT /reviews/{id}\*\*: Updates an existing review.  
5. \*\*DELETE /reviews/{id}\*\*: Deletes a review.  
  
Note that this is just one way to implement these views, there might be other ways depending on how you want to structure your code.  
  
I hope it helps! Let me know if you have any questions or need further assistance.

Here is the views.py file for CRUD operations on the Reviews Model:  
  
```python  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.parsers import JSONParser  
from rest\_framework.renderers import JSONRenderer  
from.models import Review  
from.serializers import ReviewSerializer  
import json  
# Create your views here.  
  
  
class ReviewList(APIView):  
  
 # GET method to get all reviews  
 def get(self,request):  
 review=Review.objects.all()  
 serializer=ReviewSerializer(review,many=True)  
 return Response(serializer.data)  
  
  
 # POST method to create new review  
 def post(self,request):  
 data=request.data  
 serializer=ReviewSerializer(data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'review added'})  
 else:  
 return Response(serializer.errors)  
  
  
class ReviewDetail(APIView):  
  
 # GET method to retrieve specific review details  
 def get\_object(self,id):  
 try:  
 return Review.objects.get(id=id)  
 except Exception as e:  
 return None  
  
 def get(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 serializer=ReviewSerializer(review)  
 return Response(serializer.data)  
 else:  
 return Response({'msg':'no record found'})  
  
 # PUT method to update existing review  
 def put(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 data=request.data  
 serializer=ReviewSerializer(review,data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'updated successfully'})  
 else:  
 return Response(serializer.errors)  
 else:  
 return Response({'msg':'no record found'})  
  
  
  
 # DELETE method to delete review  
 def delete(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 review.delete()  
 return Response({'msg':'deleted successfully'})  
 else:  
 return Response({'msg':'no record found'})  
```  
  
This view handles the following actions:  
1. \*\*GET /reviews\*\*: Returns a list of all reviews.  
2. \*\*POST /reviews\*\*: Creates a new review.  
3. \*\*GET /reviews/{id}\*\*: Retrieves the details of a specific review.  
4. \*\*PUT /reviews/{id}\*\*: Updates an existing review.  
5. \*\*DELETE /reviews/{id}\*\*: Deletes a review.  
  
Note that this is just one way to implement these views, there might be other ways depending on how you want to structure your code.  
  
I hope it helps! Let me know if you have any questions or need further assistance.

Here is the views.py file for CRUD operations on the Reviews Model:  
  
```python  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from rest\_framework.parsers import JSONParser  
from rest\_framework.renderers import JSONRenderer  
from.models import Review  
from.serializers import ReviewSerializer  
import json  
# Create your views here.  
  
  
class ReviewList(APIView):  
  
 # GET method to get all reviews  
 def get(self,request):  
 review=Review.objects.all()  
 serializer=ReviewSerializer(review,many=True)  
 return Response(serializer.data)  
  
  
 # POST method to create new review  
 def post(self,request):  
 data=request.data  
 serializer=ReviewSerializer(data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'review added'})  
 else:  
 return Response(serializer.errors)  
  
  
class ReviewDetail(APIView):  
  
 # GET method to retrieve specific review details  
 def get\_object(self,id):  
 try:  
 return Review.objects.get(id=id)  
 except Exception as e:  
 return None  
  
 def get(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 serializer=ReviewSerializer(review)  
 return Response(serializer.data)  
 else:  
 return Response({'msg':'no record found'})  
  
 # PUT method to update existing review  
 def put(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 data=request.data  
 serializer=ReviewSerializer(review,data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response({'msg':'updated successfully'})  
 else:  
 return Response(serializer.errors)  
 else:  
 return Response({'msg':'no record found'})  
  
  
  
 # DELETE method to delete review  
 def delete(self,request,id):  
 review=self.get\_object(id=id)  
 if review is not None:  
 review.delete()  
 return Response({'msg':'deleted successfully'})  
 else:  
 return Response({'msg':'no record found'})  
```  
  
This view handles the following actions:  
1. \*\*GET /reviews\*\*: Returns a list of all reviews.  
2. \*\*POST /reviews\*\*: Creates a new review.  
3. \*\*GET /reviews/{id}\*\*: Retrieves the details of a specific review.  
4. \*\*PUT /reviews/{id}\*\*: Updates an existing review.  
5. \*\*DELETE /reviews/{id}\*\*: Deletes a review.  
  
Note that this is just one way to implement these views, there might be other ways depending on how you want to structure your code.  
  
I hope it helps! Let me know if you have any questions or need further assistance.