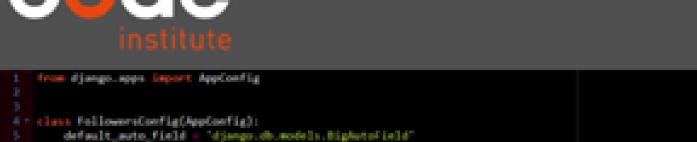
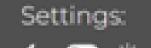


mane - "followers"

CI Python Linter











Results: All clear, no errors found



CI Python Linter

```
ros diango de isport models.
 ros diango.contrib.auth.models import User
class Follower(models . Model):
    follower model, related to 'owner' and 'followed'.
    'owner' is a topy that is following a treat-
    "followed" is a User that is followed by 'owner'.
    'unique together' makes cure a user can't 'double follow' the same user.
    owner - models.foreignKey(
        User, related name: following, on deletermodels.CASCADE
    followed = models.foreignKey(
        User, related_name='followed', on_delate=models.CASCASE
    created_st = models.Outsfirefield(outs_now_add-line)
    Charte Bedar
        ordering = ['-created_at']
        unique together : ['outer', 'followed']
    dof __str_(salf):
        return f'(self.owner) (self.followed)'
```

Settings:



Results:

All clear, no errors found

```
row diseas.db import Integrityferor
from rest framework import serializers
from .models import follower
(lass FollowerSerializer(serializers, ModelSerializer)
   torializer for the follower model.
   Create method handles the unique constraint on "owner" and "followed".
   owner = serializers.ReadOnlyfield(source:'owner.username')
   followed name - perializers. ReadOnlyField(source-followed_username')
   class Betail
       model = Follower
       finists - I
           'id', 'owner', 'created st', 'followed, 'followed name'
   def create(self, validated data):
           return seper().create(validated_data)
       except Integrityforor:
           raise serializers. Walldetlosteror(('detail': 'possible duplicate'))
```

Settings:



Results:

All clear, no errors found





CI Python Linter

```
from rest_framework import generics, permissions
from cookbook and permissions import InDanarOrfoadOnly
from .models import follower
from , serializers import Followerterializer
class followerList(generics_ListCreateMTView):
   List all followers, i.e. all instances of a user
   following another uper'.
   Create a follower, i.e. follow a user if lagged in.
   Perfore create: associate the current logged in user with a follower.
   permission_classes = [permissions.fuAuthenticatedOrReadOnly]
   overwest = follower.objects.all()
   perializer class - followerierializer
   def perform create(self, serializer):
       perializer.paye(ownerself.request.waer)
class FollowerOstail(generics.SetrieveDestroyAFSView);
   Batteliana a follower
   No Update view, as we either follow or unfollow users
   Destroy a follower, i.e. unfollow someone if owner
   permission_classes = [IsOwnerOrReadOnly]
   queryset = follower.objects.all()
   certalizer class - followertertalizer
```

Settings:



Results:

All clear, no errors found