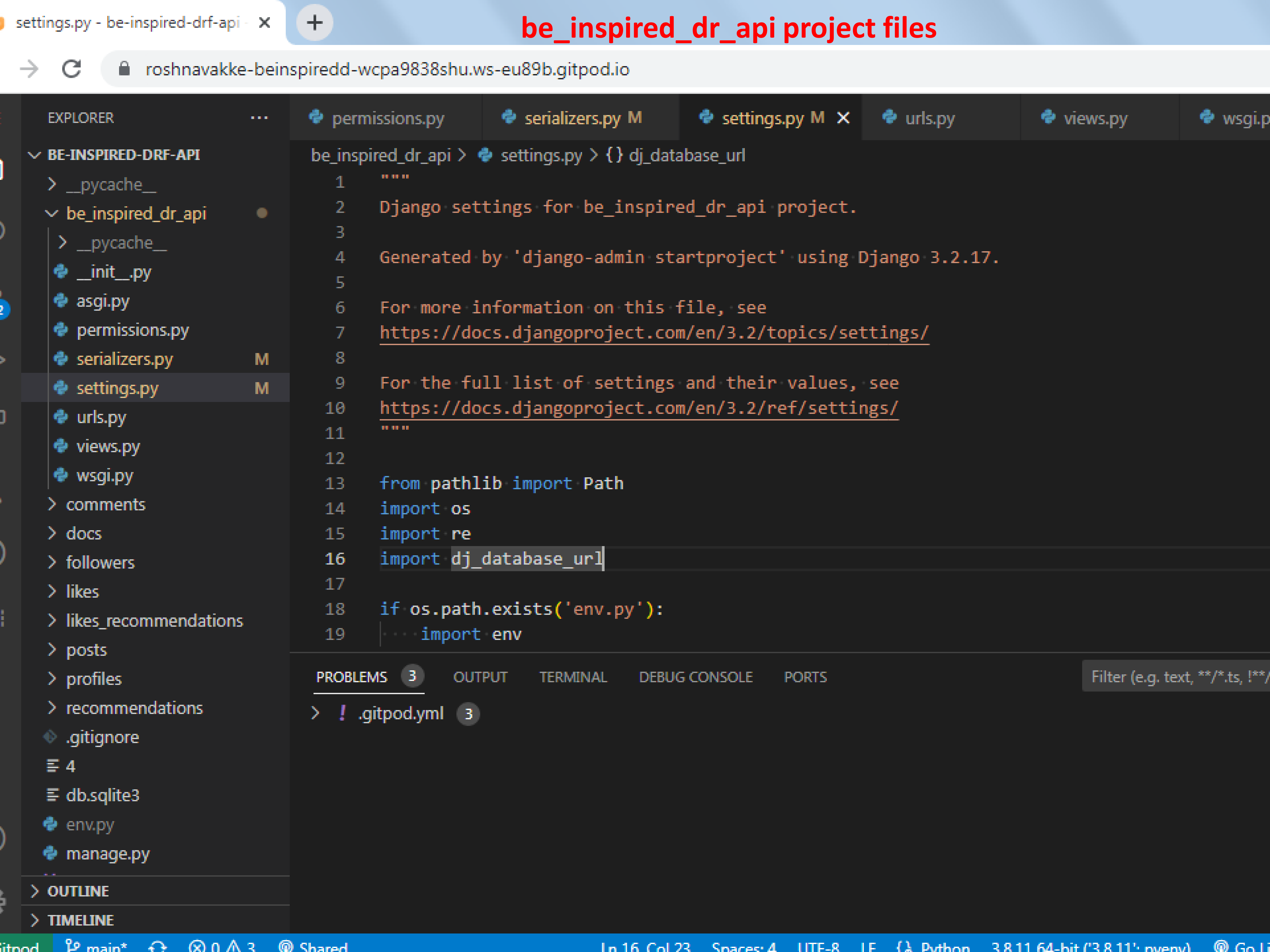
be_inspired_dr_api project files



settings.py - be-inspired-drf-api

roshnavakke-beinspiredd-wcpa9838shu.ws-eu89b.gitpod.io

EXPLORER

BE-INSPIRED-DRF-API
- __pycache__
- be_inspired_dr_api
  - __pycache__
  - __init__.py
  - asgi.py
  - permissions.py
  - serializers.py M
  - settings.py M
  - urls.py
  - views.py
  - wsgi.py
- comments
- docs
- followers
- likes
- likes_recommendations
- posts
- profiles
- recommendations
- .gitignore
- 4
- db.sqlite3
- env.py
- manage.py

OUTLINE

TIMELINE

permissions.py | serializers.py M | settings.py M | urls.py | views.py | wsgi.p

be_inspired_dr_api > settings.py > {} dj_database_url

```python
"""
Django settings for be_inspired_dr_api project.

Generated by 'django-admin startproject' using Django 3.2.17.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

from pathlib import Path
import os
import re
import dj_database_url

if os.path.exists('env.py'):
    import env
```

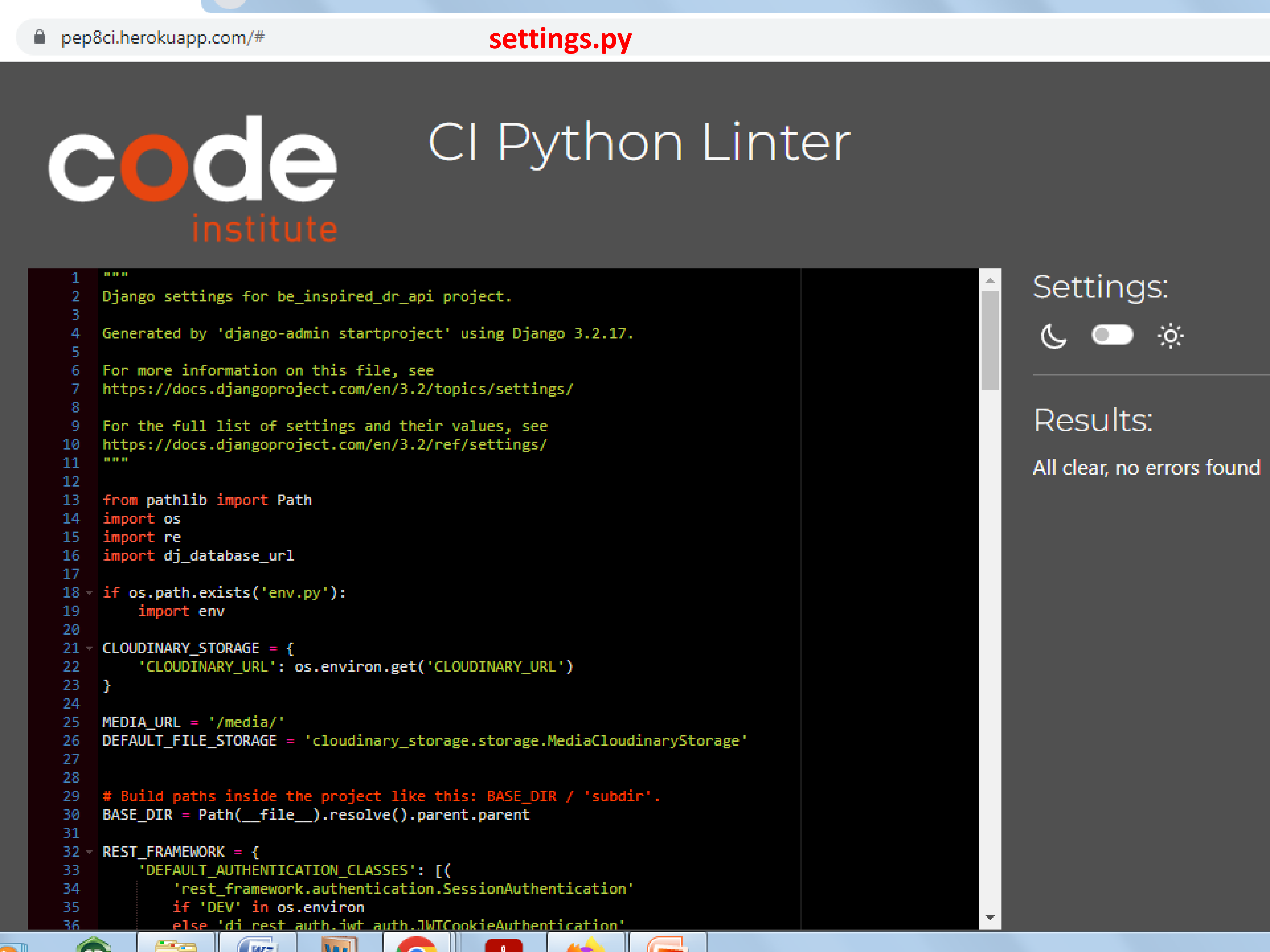
PROBLEMS 3 | OUTPUT | TERMINAL | DEBUG CONSOLE | PORTS

Filter (e.g. text, **/*.ts, !**/

! .gitpod.yml 3

Gitpod | main* | ⊗ 0 ⚠ 3 | Shared | Ln 16, Col 23 | Spaces: 4 | UTF-8 | LF | {} Python | 3.8.11 64-bit ('3.8.11': pyenv) | Go Li pep8ci.herokuapp.com/#

settings.py



CI Python Linter

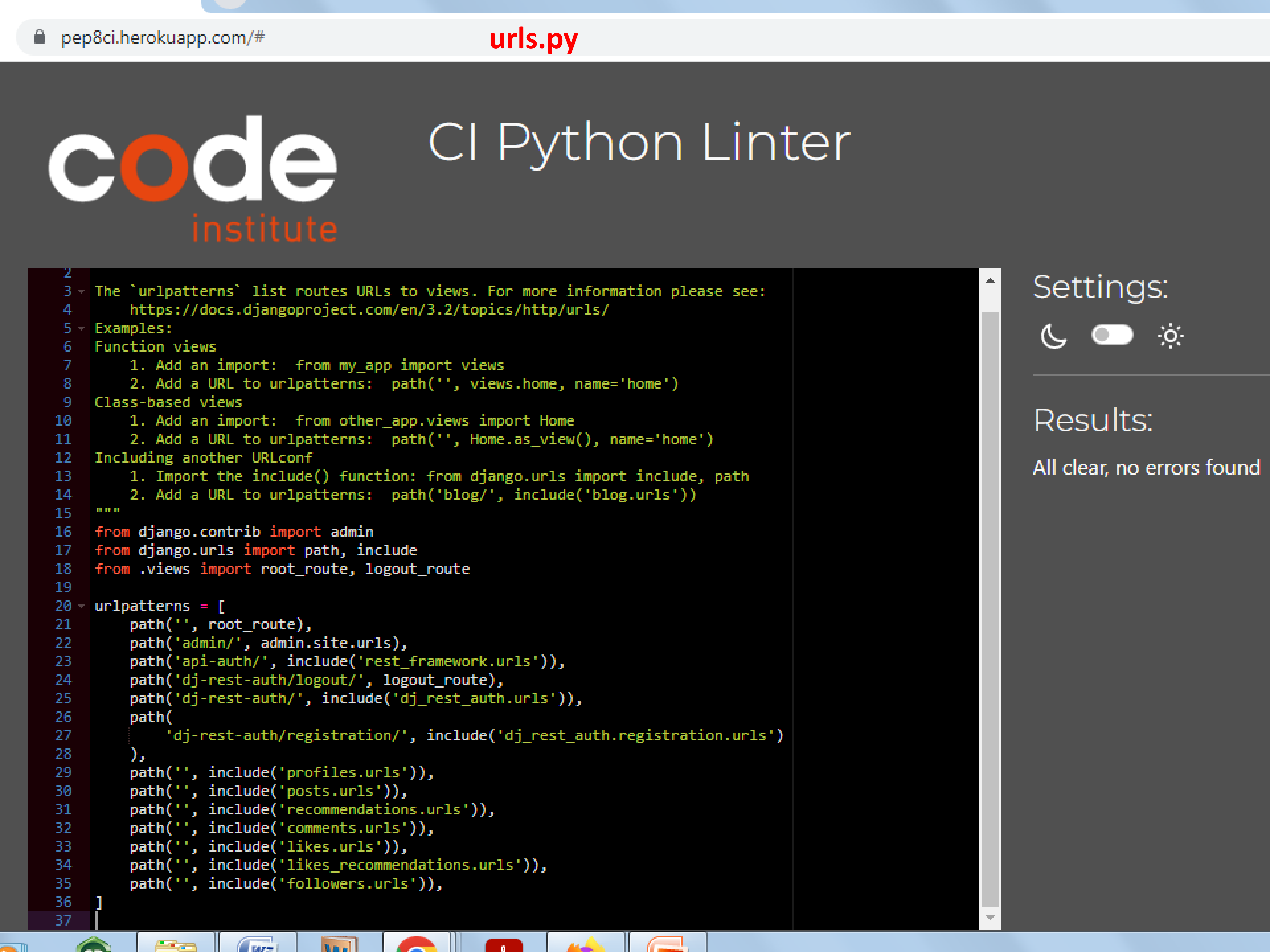
```
"""
Django settings for be_inspired_dr_api project.

Generated by 'django-admin startproject' using Django 3.2.17.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

from pathlib import Path
import os
import re
import dj_database_url

if os.path.exists('env.py'):
    import env

CLOUDINARY_STORAGE = {
    'CLOUDINARY_URL': os.environ.get('CLOUDINARY_URL')
}

MEDIA_URL = '/media/'
DEFAULT_FILE_STORAGE = 'cloudinary_storage.storage.MediaCloudinaryStorage'


# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [(
        'rest_framework.authentication.SessionAuthentication'
        if 'DEV' in os.environ
        else 'dj_rest_auth.jwt_auth.JWTCookieAuthentication'
```

Settings:

Results:

All clear, no errors found

urls.py



pep8ci.herokuapp.com/#

CI Python Linter

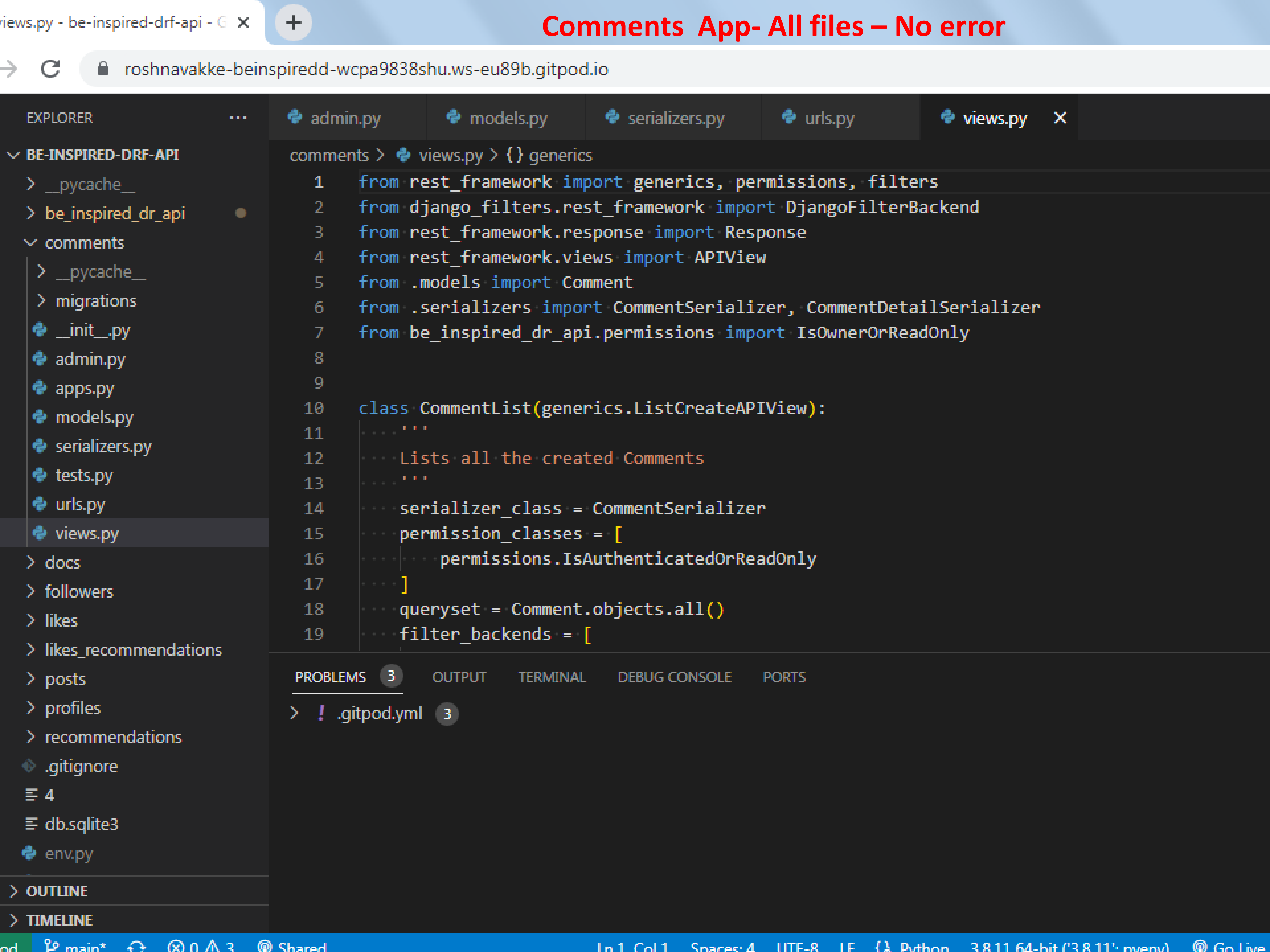
```
The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include
from .views import root_route, logout_route

urlpatterns = [
    path('', root_route),
    path('admin/', admin.site.urls),
    path('api-auth/', include('rest_framework.urls')),
    path('dj-rest-auth/logout/', logout_route),
    path('dj-rest-auth/', include('dj_rest_auth.urls')),
    path(
        'dj-rest-auth/registration/', include('dj_rest_auth.registration.urls')
    ),
    path('', include('profiles.urls')),
    path('', include('posts.urls')),
    path('', include('recommendations.urls')),
    path('', include('comments.urls')),
    path('', include('likes.urls')),
    path('', include('likes_recommendations.urls')),
    path('', include('followers.urls')),
]
```

Settings:

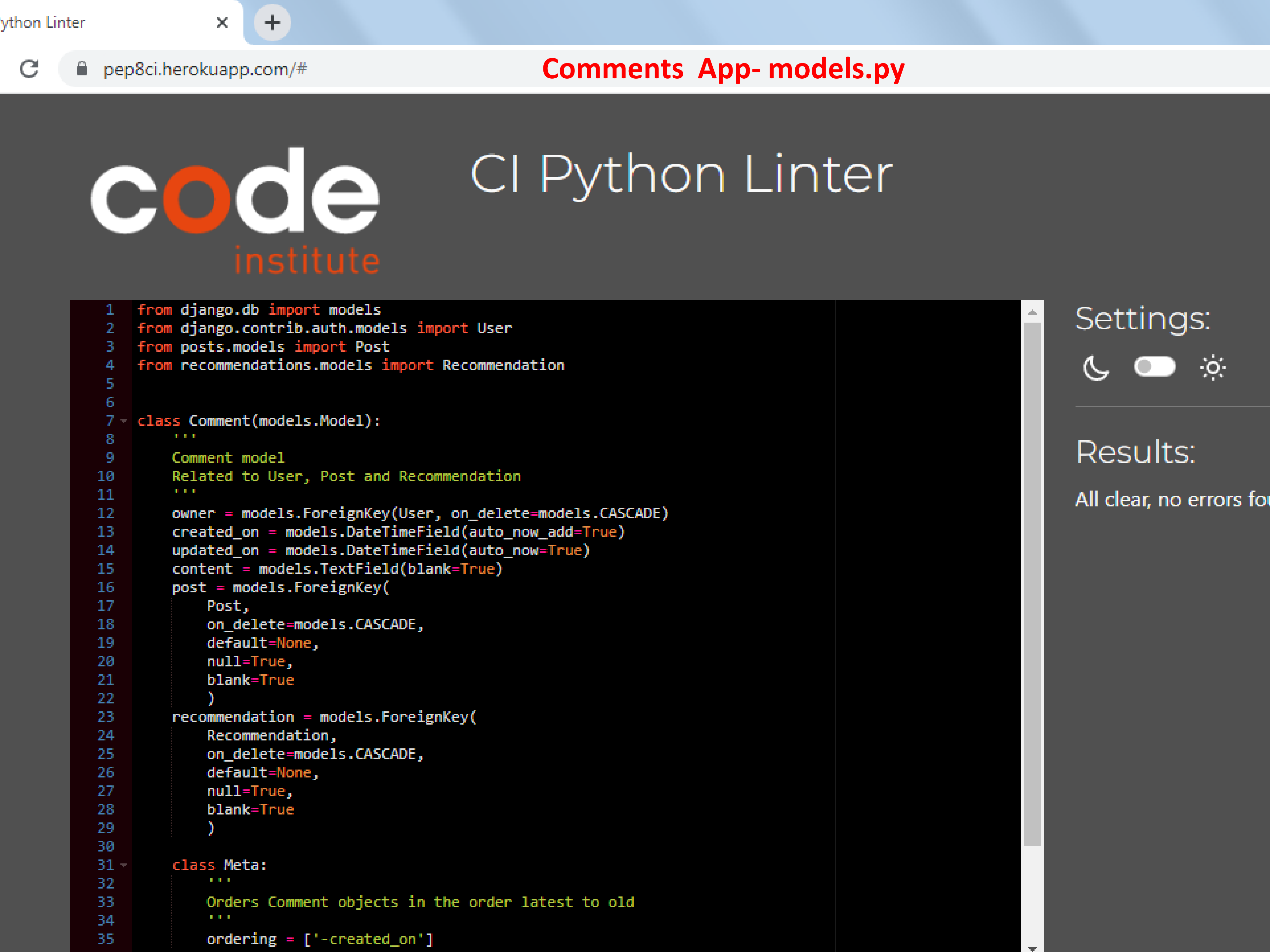
Results:

All clear, no errors found

Comments App- All files – No error



```python
from rest_framework import generics, permissions, filters
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework.response import Response
from rest_framework.views import APIView
from .models import Comment
from .serializers import CommentSerializer, CommentDetailSerializer
from be_inspired_dr_api.permissions import IsOwnerOrReadOnly


class CommentList(generics.ListCreateAPIView):
    '''
    Lists all the created Comments
    '''
    serializer_class = CommentSerializer
    permission_classes = [
        permissions.IsAuthenticatedOrReadOnly
    ]
    queryset = Comment.objects.all()
    filter_backends = [
```



ython Linter

pep8ci.herokuapp.com/#

Comments App- models.py

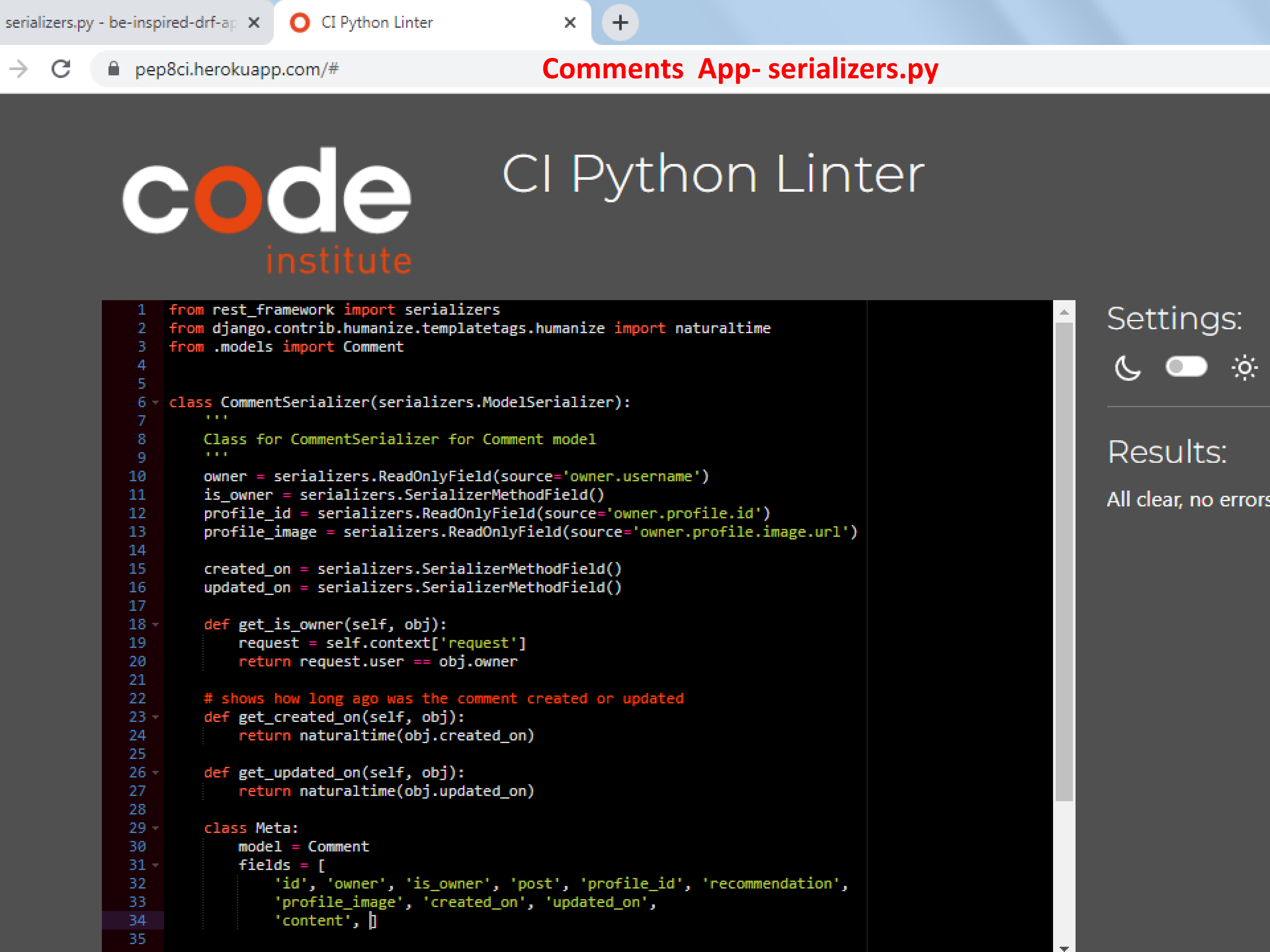
CI Python Linter

```python
from django.db import models
from django.contrib.auth.models import User
from posts.models import Post
from recommendations.models import Recommendation


class Comment(models.Model):
    '''
    Comment model
    Related to User, Post and Recommendation
    '''
    owner = models.ForeignKey(User, on_delete=models.CASCADE)
    created_on = models.DateTimeField(auto_now_add=True)
    updated_on = models.DateTimeField(auto_now=True)
    content = models.TextField(blank=True)
    post = models.ForeignKey(
        Post,
        on_delete=models.CASCADE,
        default=None,
        null=True,
        blank=True
        )
    recommendation = models.ForeignKey(
        Recommendation,
        on_delete=models.CASCADE,
        default=None,
        null=True,
        blank=True
        )

    class Meta:
        '''
        Orders Comment objects in the order latest to old
        '''
        ordering = ['-created_on']
```

Settings:

Results:

All clear, no errors fo

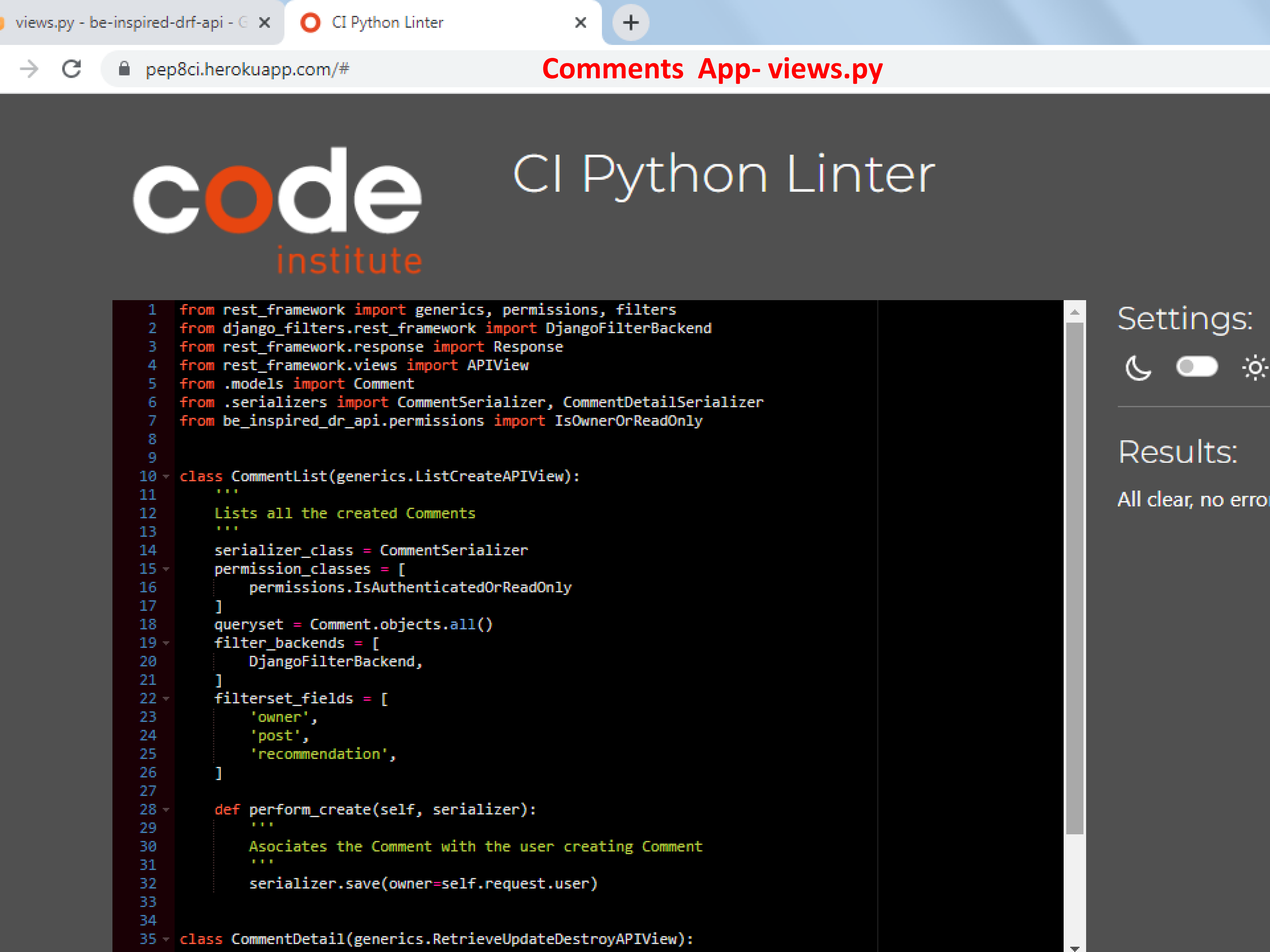

Comments App- serializers.py

```python
from rest_framework import serializers
from django.contrib.humanize.templatetags.humanize import naturaltime
from .models import Comment


class CommentSerializer(serializers.ModelSerializer):
    '''
    Class for CommentSerializer for Comment model
    '''
    owner = serializers.ReadOnlyField(source='owner.username')
    is_owner = serializers.SerializerMethodField()
    profile_id = serializers.ReadOnlyField(source='owner.profile.id')
    profile_image = serializers.ReadOnlyField(source='owner.profile.image.url')

    created_on = serializers.SerializerMethodField()
    updated_on = serializers.SerializerMethodField()

    def get_is_owner(self, obj):
        request = self.context['request']
        return request.user == obj.owner

    # shows how long ago was the comment created or updated
    def get_created_on(self, obj):
        return naturaltime(obj.created_on)

    def get_updated_on(self, obj):
        return naturaltime(obj.updated_on)

    class Meta:
        model = Comment
        fields = [
            'id', 'owner', 'is_owner', 'post', 'profile_id', 'recommendation',
            'profile_image', 'created_on', 'updated_on',
            'content', ]

```

Results:

All clear, no errors

Comments App- views.py



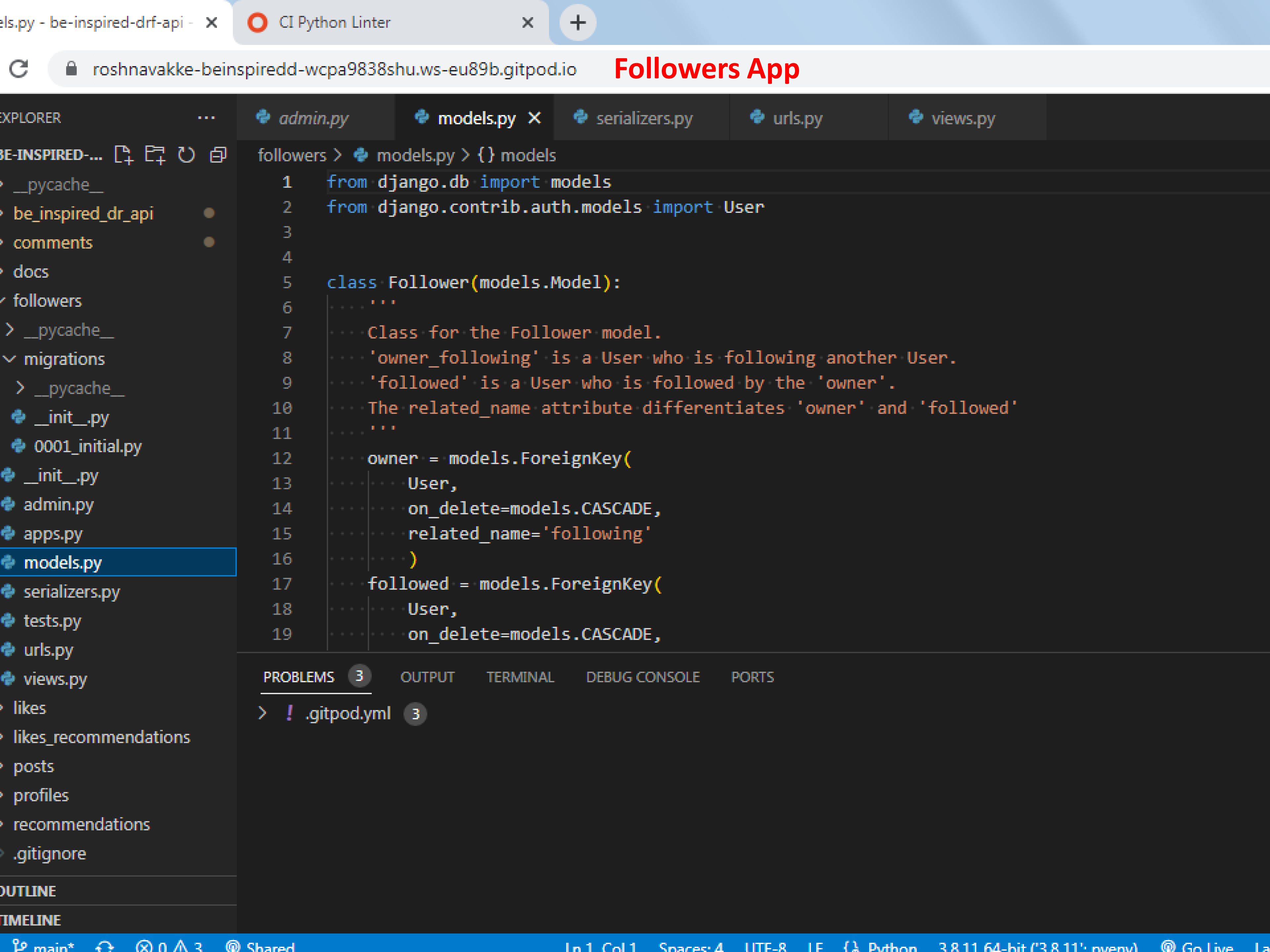
CI Python Linter

```
from rest_framework import generics, permissions, filters
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework.response import Response
from rest_framework.views import APIView
from .models import Comment
from .serializers import CommentSerializer, CommentDetailSerializer
from be_inspired_dr_api.permissions import IsOwnerOrReadOnly


class CommentList(generics.ListCreateAPIView):
    '''
    Lists all the created Comments
    '''
    serializer_class = CommentSerializer
    permission_classes = [
        permissions.IsAuthenticatedOrReadOnly
    ]
    queryset = Comment.objects.all()
    filter_backends = [
        DjangoFilterBackend,
    ]
    filterset_fields = [
        'owner',
        'post',
        'recommendation',
    ]

    def perform_create(self, serializer):
        '''
        Asociates the Comment with the user creating Comment
        '''
        serializer.save(owner=self.request.user)


class CommentDetail(generics.RetrieveUpdateDestroyAPIView):
```

Settings:

Results:

All clear, no erro



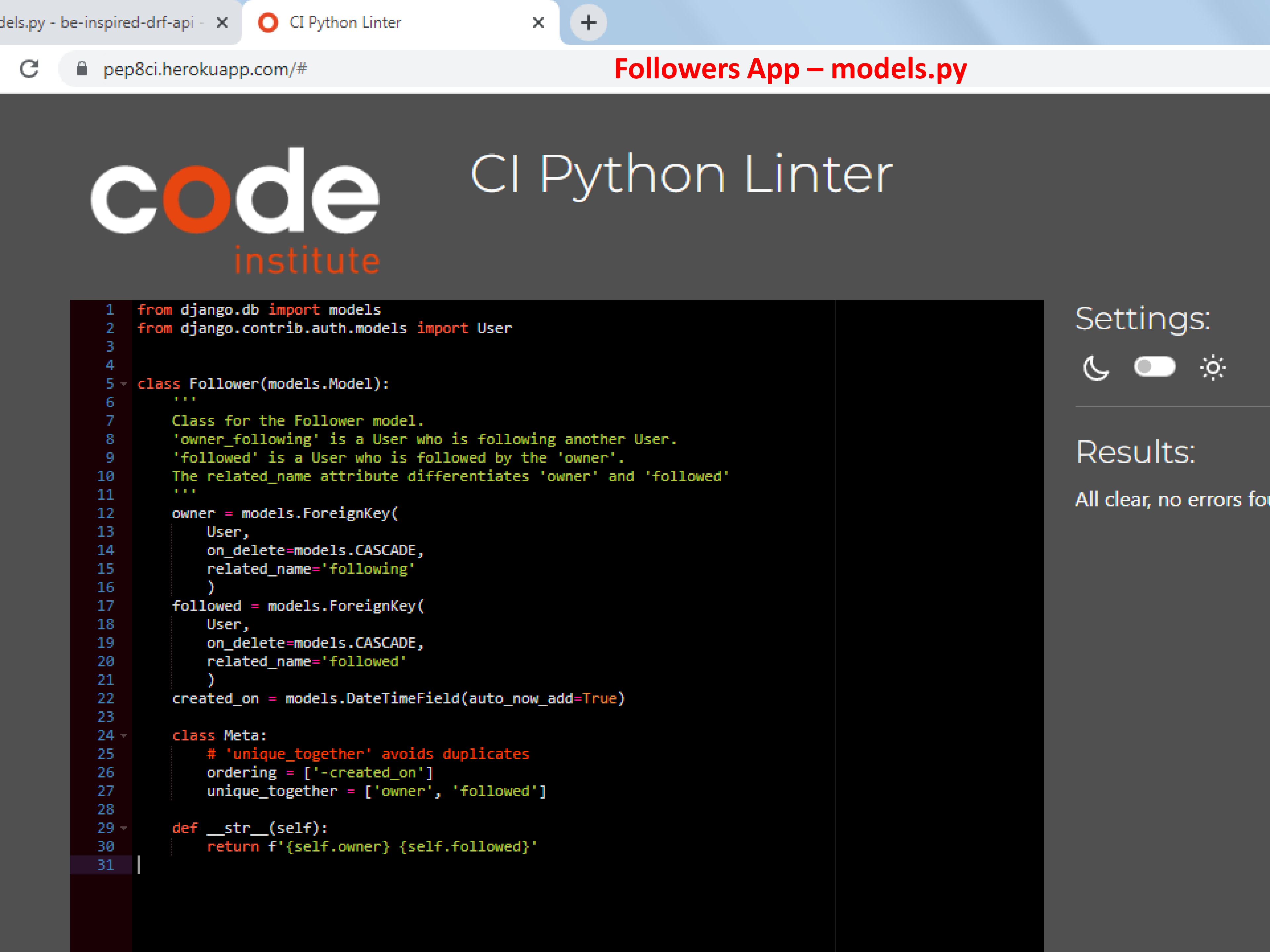
els.py - be-inspired-drf-api | CI Python Linter

roshnavakke-beinspiredd-wcpa9838shu.ws-eu89b.gitpod.io **Followers App**

EXPLORER | admin.py | models.py | serializers.py | urls.py | views.py

followers > models.py > {} models

```python
from django.db import models
from django.contrib.auth.models import User


class Follower(models.Model):
    '''
    Class for the Follower model.
    'owner_following' is a User who is following another User.
    'followed' is a User who is followed by the 'owner'.
    The related_name attribute differentiates 'owner' and 'followed'
    '''
    owner = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
        related_name='following'
        )
    followed = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
```

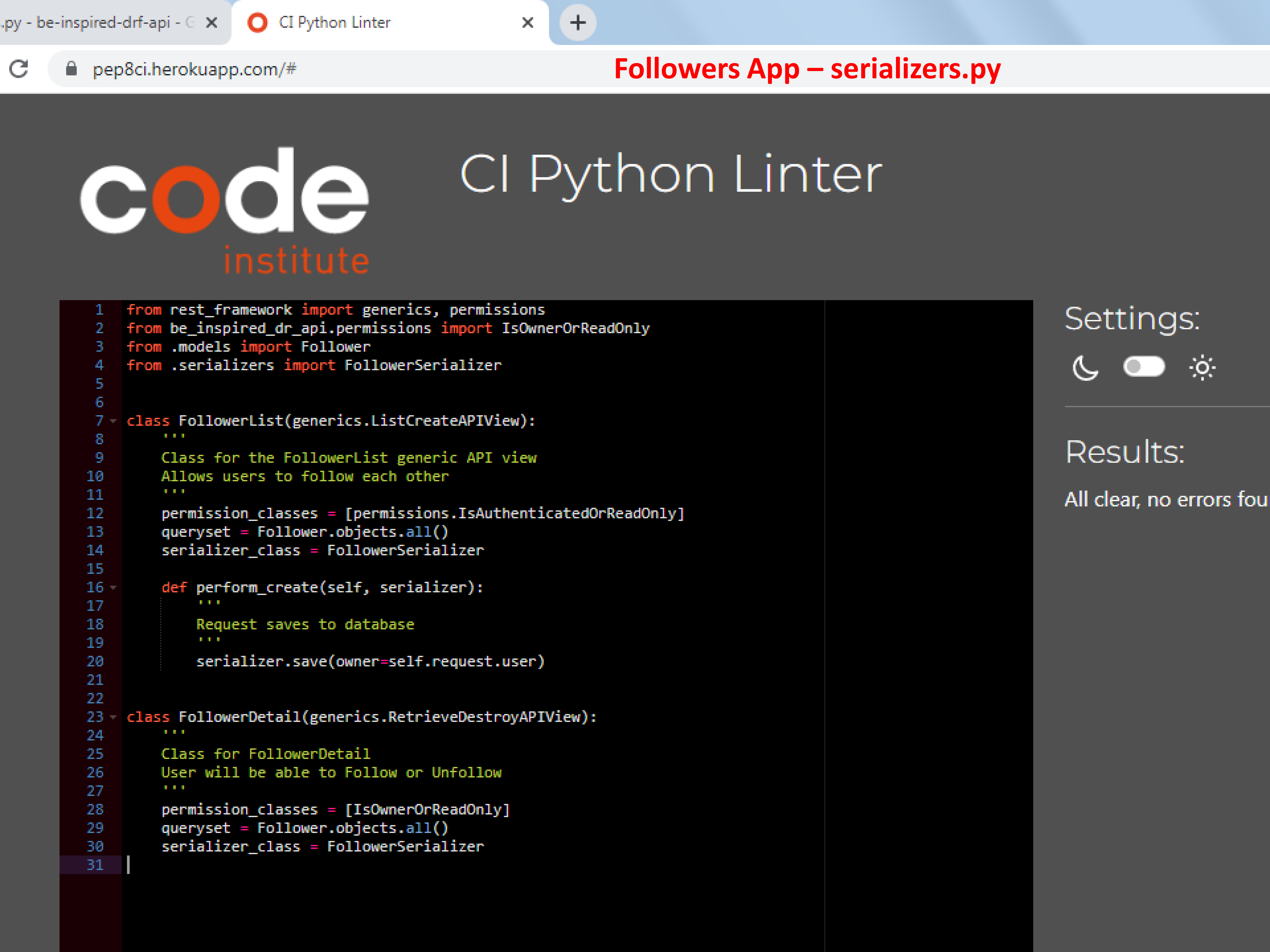


Followers App – models.py

```python
from django.db import models
from django.contrib.auth.models import User


class Follower(models.Model):
    '''
    Class for the Follower model.
    'owner_following' is a User who is following another User.
    'followed' is a User who is followed by the 'owner'.
    The related_name attribute differentiates 'owner' and 'followed'
    '''
    owner = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
        related_name='following'
        )
    followed = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
        related_name='followed'
        )
    created_on = models.DateTimeField(auto_now_add=True)

    class Meta:
        # 'unique_together' avoids duplicates
        ordering = ['-created_on']
        unique_together = ['owner', 'followed']

    def __str__(self):
        return f'{self.owner} {self.followed}'
```

Results:

All clear, no errors fo



Followers App – serializers.py

CI Python Linter

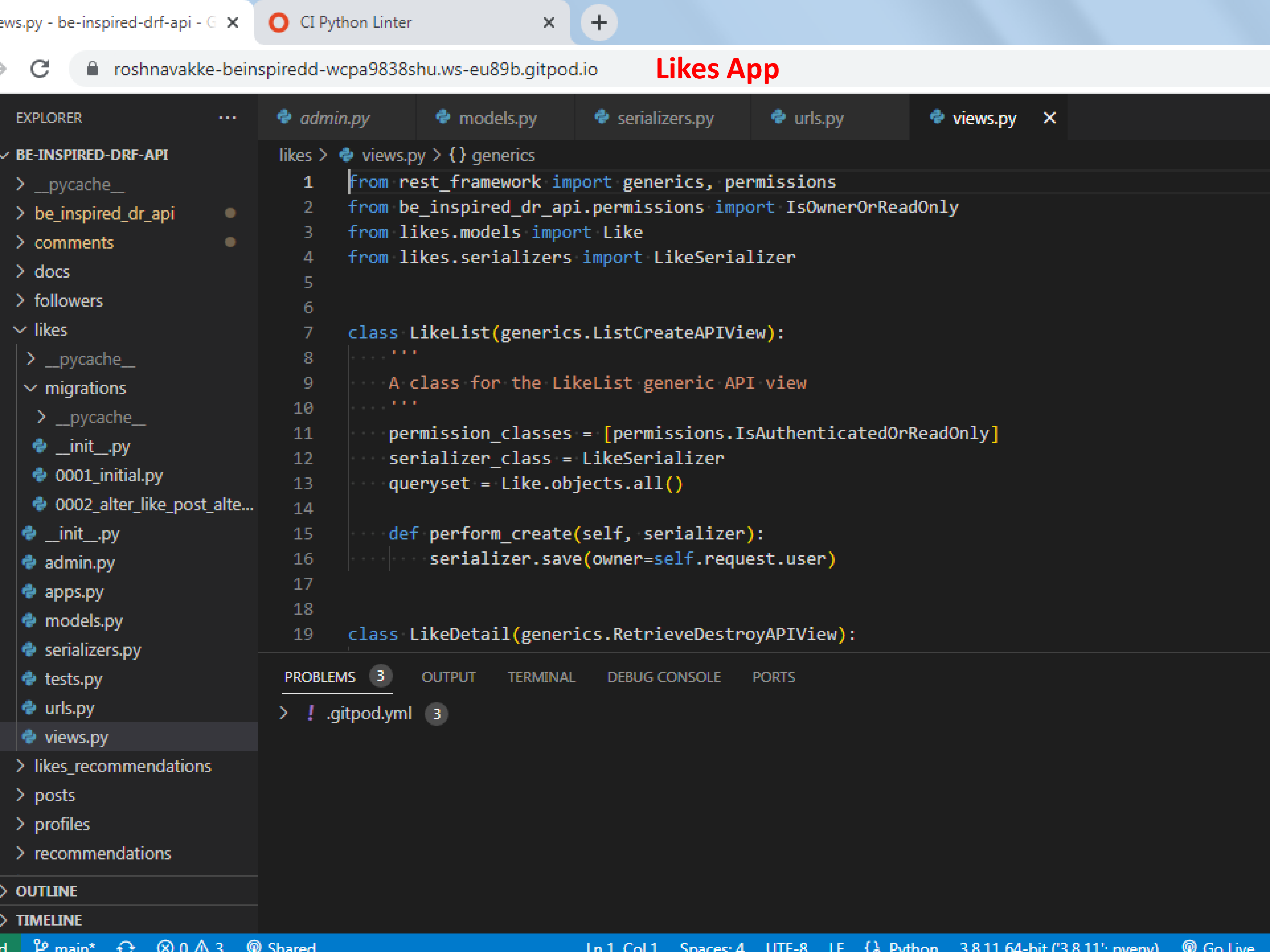
```
from rest_framework import generics, permissions
from be_inspired_dr_api.permissions import IsOwnerOrReadOnly
from .models import Follower
from .serializers import FollowerSerializer


class FollowerList(generics.ListCreateAPIView):
    '''
    Class for the FollowerList generic API view
    Allows users to follow each other
    '''
    permission_classes = [permissions.IsAuthenticatedOrReadOnly]
    queryset = Follower.objects.all()
    serializer_class = FollowerSerializer

    def perform_create(self, serializer):
        '''
        Request saves to database
        '''
        serializer.save(owner=self.request.user)


class FollowerDetail(generics.RetrieveDestroyAPIView):
    '''
    Class for FollowerDetail
    User will be able to Follow or Unfollow
    '''
    permission_classes = [IsOwnerOrReadOnly]
    queryset = Follower.objects.all()
    serializer_class = FollowerSerializer

```

Settings:

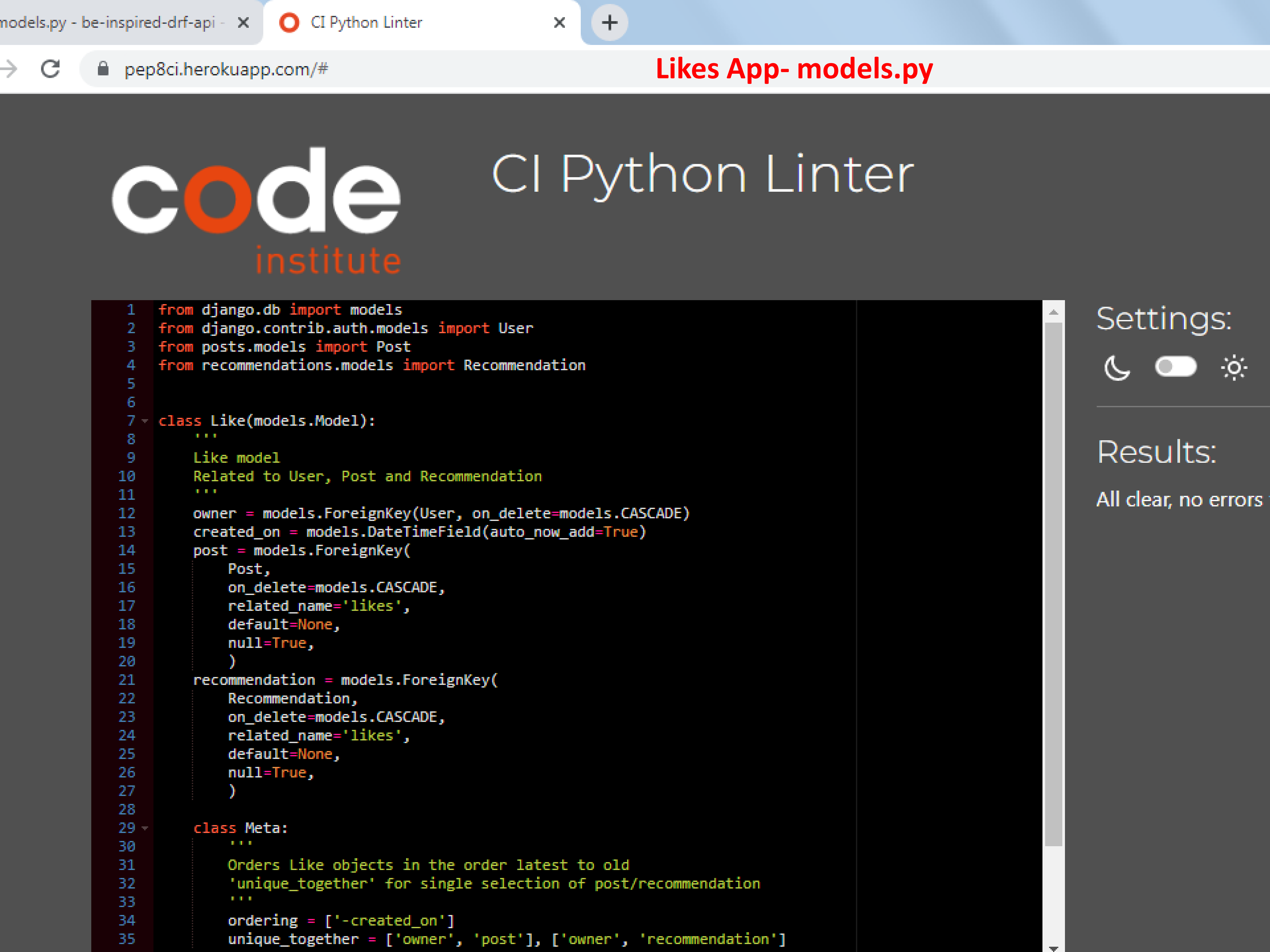
Results:

All clear, no errors fou

Likes App



```python
from rest_framework import generics, permissions
from be_inspired_dr_api.permissions import IsOwnerOrReadOnly
from likes.models import Like
from likes.serializers import LikeSerializer


class LikeList(generics.ListCreateAPIView):
    '''
    A class for the LikeList generic API view
    '''
    permission_classes = [permissions.IsAuthenticatedOrReadOnly]
    serializer_class = LikeSerializer
    queryset = Like.objects.all()

    def perform_create(self, serializer):
        serializer.save(owner=self.request.user)


class LikeDetail(generics.RetrieveDestroyAPIView):
```



models.py - be-inspired-drf-api ×   CI Python Linter ×

pep8ci.herokuapp.com/#

Likes App- models.py

CI Python Linter

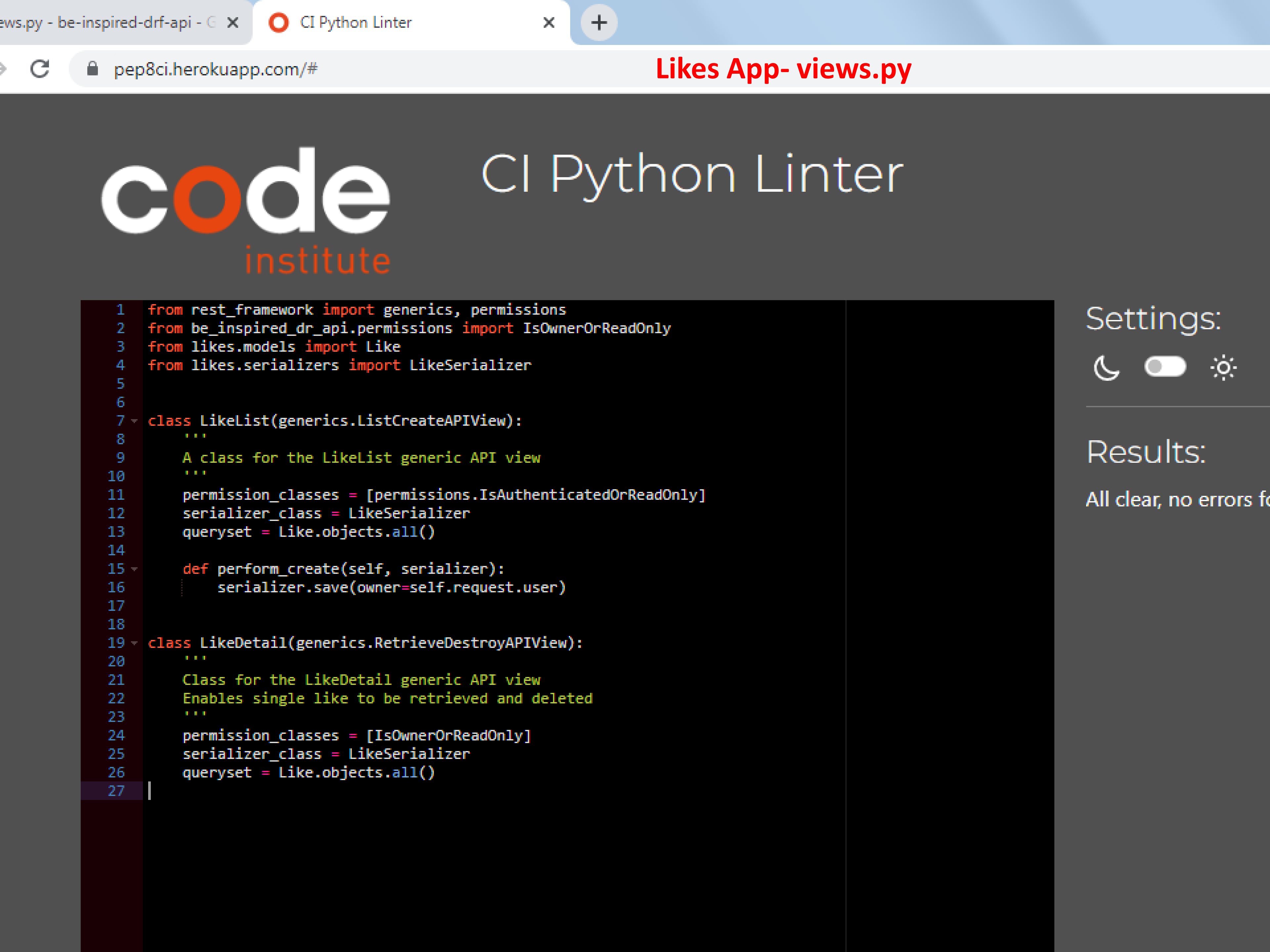
```python
from django.db import models
from django.contrib.auth.models import User
from posts.models import Post
from recommendations.models import Recommendation


class Like(models.Model):
    '''
    Like model
    Related to User, Post and Recommendation
    '''
    owner = models.ForeignKey(User, on_delete=models.CASCADE)
    created_on = models.DateTimeField(auto_now_add=True)
    post = models.ForeignKey(
        Post,
        on_delete=models.CASCADE,
        related_name='likes',
        default=None,
        null=True,
        )
    recommendation = models.ForeignKey(
        Recommendation,
        on_delete=models.CASCADE,
        related_name='likes',
        default=None,
        null=True,
        )

    class Meta:
        '''
        Orders Like objects in the order latest to old
        'unique_together' for single selection of post/recommendation
        '''
        ordering = ['-created_on']
        unique_together = ['owner', 'post'], ['owner', 'recommendation']
```

Settings:

Results:

All clear, no errors

Likes App- views.py



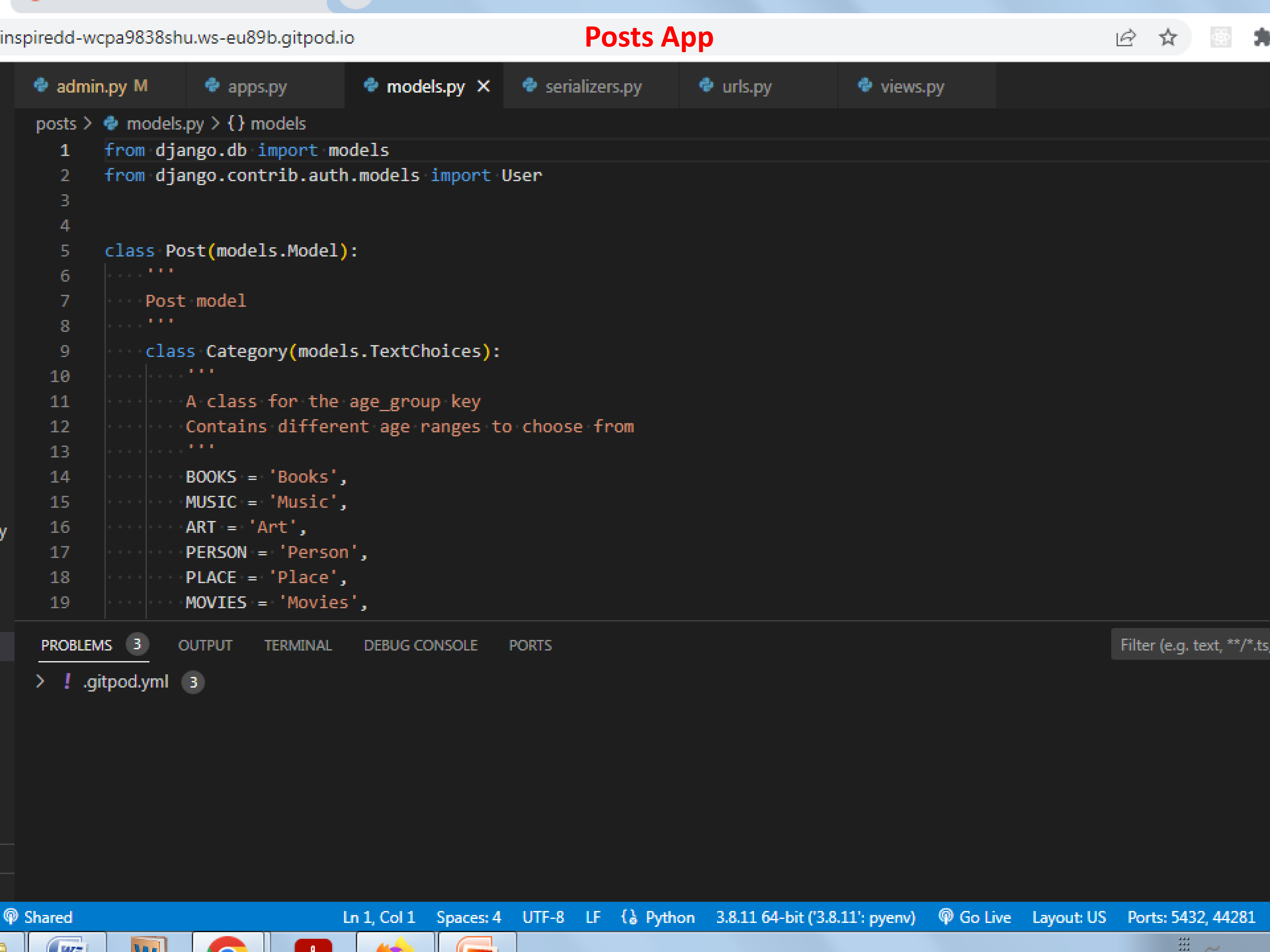
CI Python Linter

```
from rest_framework import generics, permissions
from be_inspired_dr_api.permissions import IsOwnerOrReadOnly
from likes.models import Like
from likes.serializers import LikeSerializer


class LikeList(generics.ListCreateAPIView):
    '''
    A class for the LikeList generic API view
    '''
    permission_classes = [permissions.IsAuthenticatedOrReadOnly]
    serializer_class = LikeSerializer
    queryset = Like.objects.all()

    def perform_create(self, serializer):
        serializer.save(owner=self.request.user)


class LikeDetail(generics.RetrieveDestroyAPIView):
    '''
    Class for the LikeDetail generic API view
    Enables single like to be retrieved and deleted
    '''
    permission_classes = [IsOwnerOrReadOnly]
    serializer_class = LikeSerializer
    queryset = Like.objects.all()
```

Settings:

Results:

All clear, no errors fo

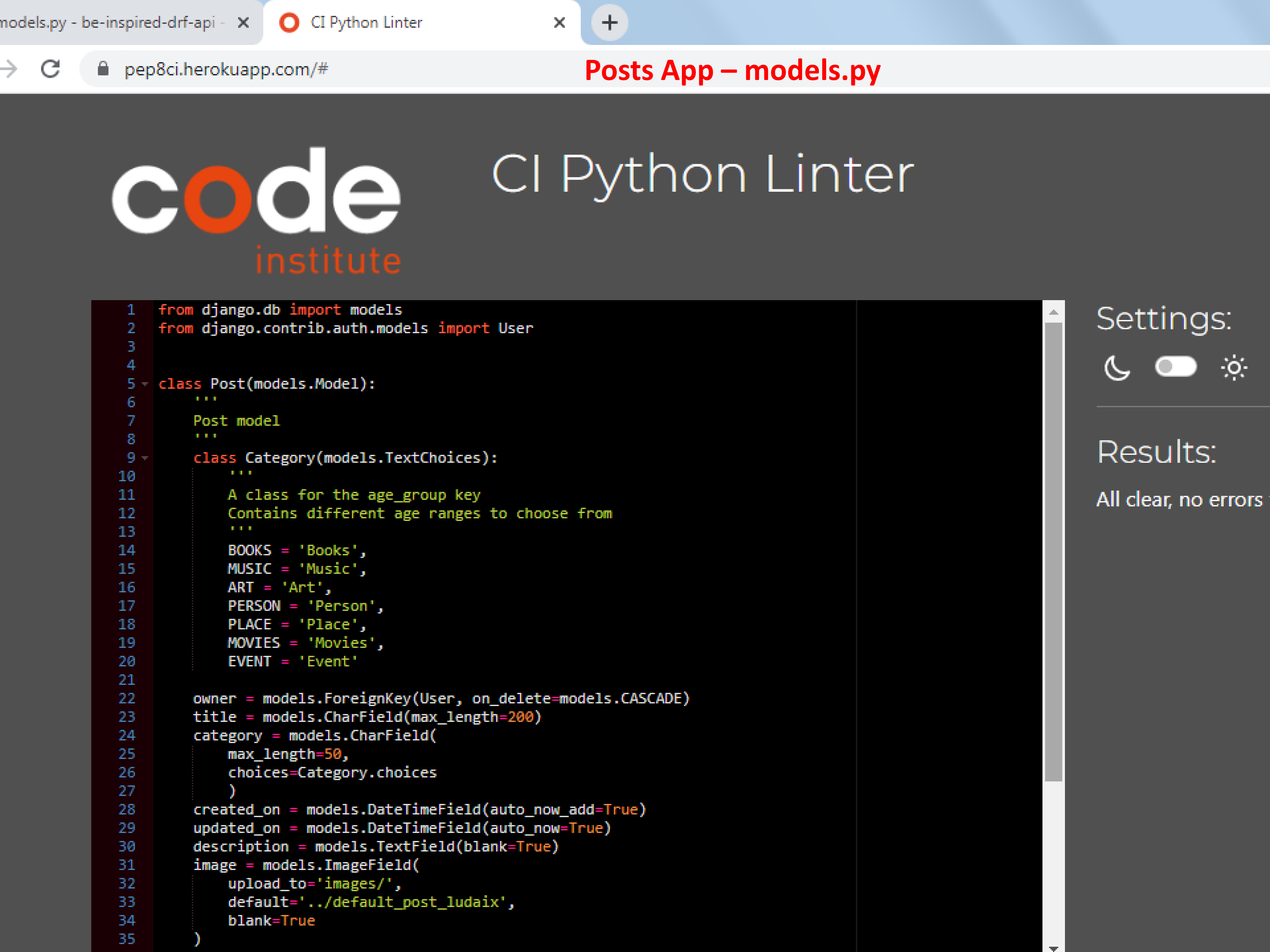


inspiredd-wcpa9838shu.ws-eu89b.gitpod.io

Posts App

```python
from django.db import models
from django.contrib.auth.models import User


class Post(models.Model):
    '''
    Post model
    '''
    class Category(models.TextChoices):
        '''
        A class for the age_group key
        Contains different age ranges to choose from
        '''
        BOOKS = 'Books',
        MUSIC = 'Music',
        ART = 'Art',
        PERSON = 'Person',
        PLACE = 'Place',
        MOVIES = 'Movies',
```

Posts App – models.py

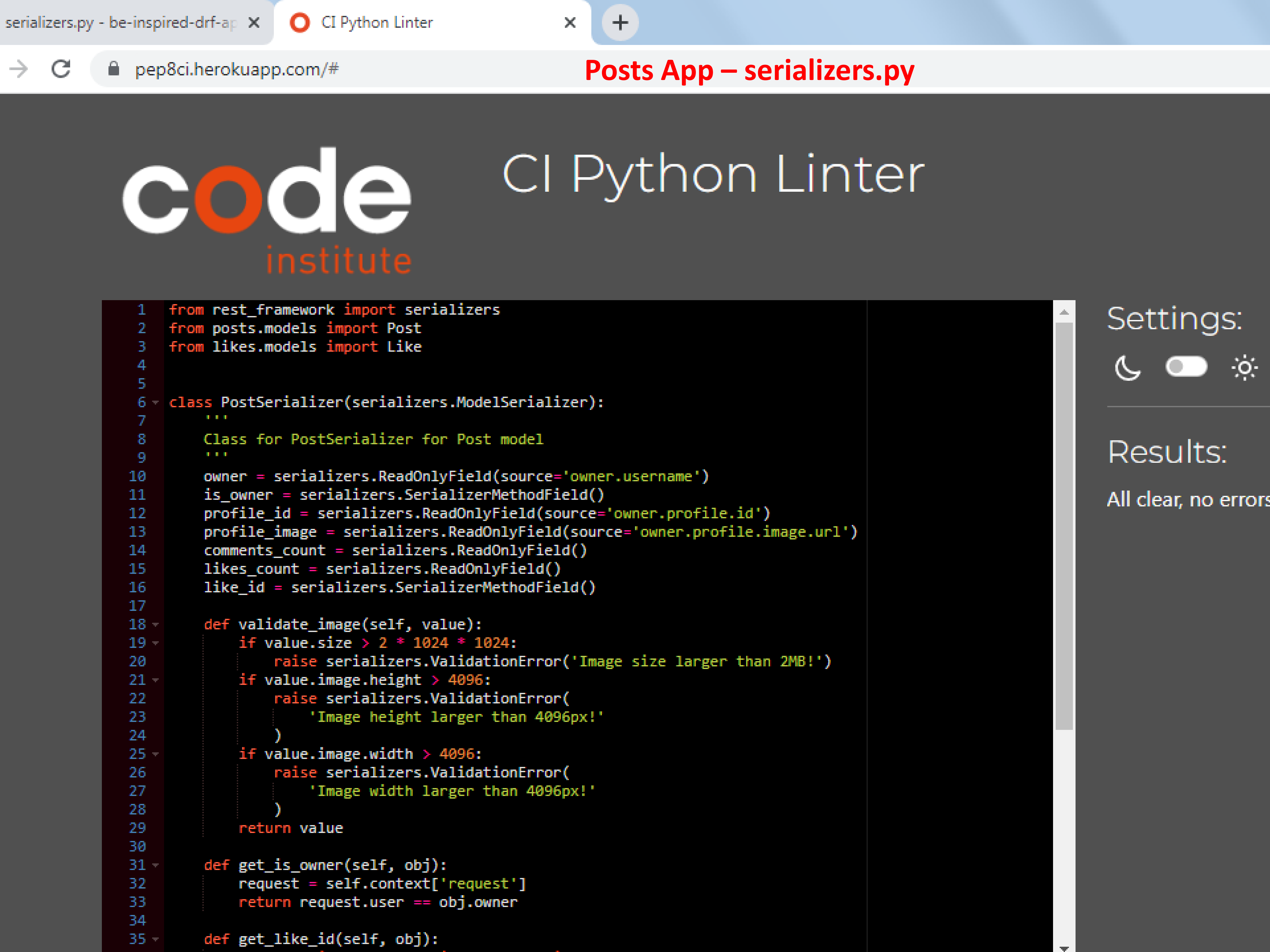


```
from django.db import models
from django.contrib.auth.models import User


class Post(models.Model):
    '''
    Post model
    '''
    class Category(models.TextChoices):
        '''
        A class for the age_group key
        Contains different age ranges to choose from
        '''
        BOOKS = 'Books',
        MUSIC = 'Music',
        ART = 'Art',
        PERSON = 'Person',
        PLACE = 'Place',
        MOVIES = 'Movies',
        EVENT = 'Event'

    owner = models.ForeignKey(User, on_delete=models.CASCADE)
    title = models.CharField(max_length=200)
    category = models.CharField(
        max_length=50,
        choices=Category.choices
        )
    created_on = models.DateTimeField(auto_now_add=True)
    updated_on = models.DateTimeField(auto_now=True)
    description = models.TextField(blank=True)
    image = models.ImageField(
        upload_to='images/',
        default='../default_post_ludaix',
        blank=True
    )
```

Results:

All clear, no errors

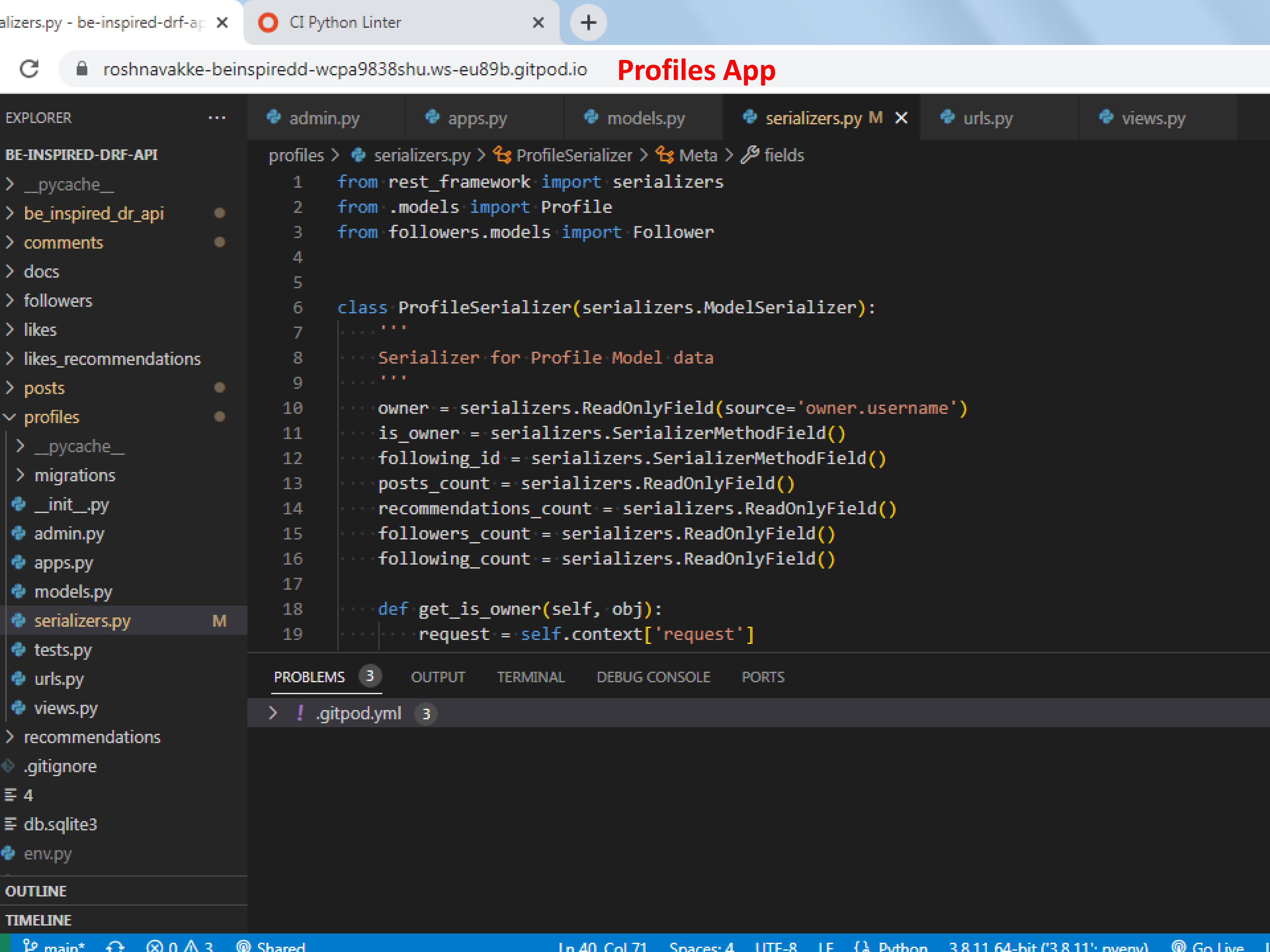
Posts App – serializers.py



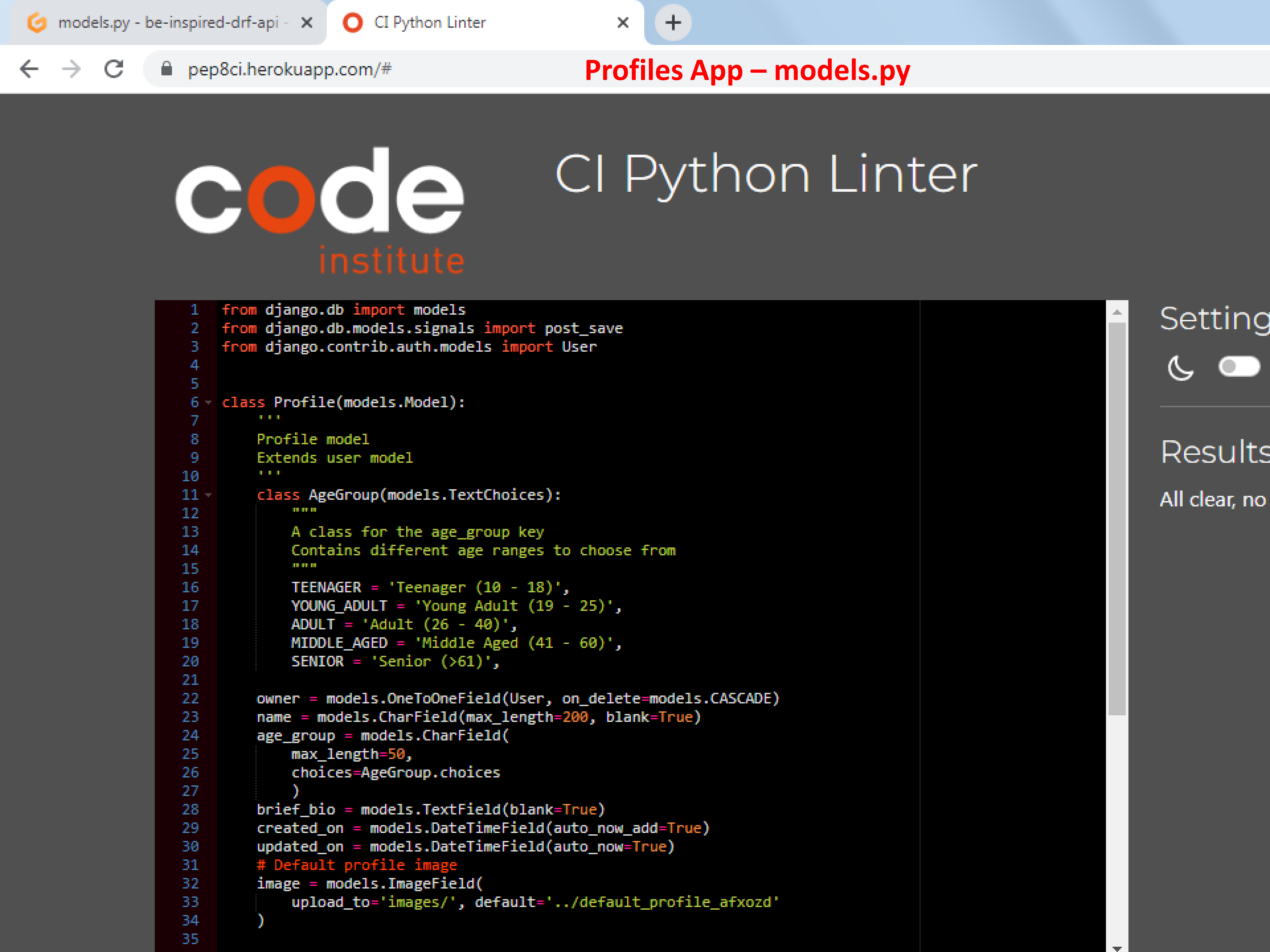
```
from rest_framework import serializers
from posts.models import Post
from likes.models import Like


class PostSerializer(serializers.ModelSerializer):
    '''
    Class for PostSerializer for Post model
    '''
    owner = serializers.ReadOnlyField(source='owner.username')
    is_owner = serializers.SerializerMethodField()
    profile_id = serializers.ReadOnlyField(source='owner.profile.id')
    profile_image = serializers.ReadOnlyField(source='owner.profile.image.url')
    comments_count = serializers.ReadOnlyField()
    likes_count = serializers.ReadOnlyField()
    like_id = serializers.SerializerMethodField()

    def validate_image(self, value):
        if value.size > 2 * 1024 * 1024:
            raise serializers.ValidationError('Image size larger than 2MB!')
        if value.image.height > 4096:
            raise serializers.ValidationError(
                'Image height larger than 4096px!'
            )
        if value.image.width > 4096:
            raise serializers.ValidationError(
                'Image width larger than 4096px!'
            )
        return value

    def get_is_owner(self, obj):
        request = self.context['request']
        return request.user == obj.owner

    def get_like_id(self, obj):
```

All clear, no errors

Profiles App



```
from rest_framework import serializers
from .models import Profile
from followers.models import Follower


class ProfileSerializer(serializers.ModelSerializer):
    '''
    Serializer for Profile Model data
    '''
    owner = serializers.ReadOnlyField(source='owner.username')
    is_owner = serializers.SerializerMethodField()
    following_id = serializers.SerializerMethodField()
    posts_count = serializers.ReadOnlyField()
    recommendations_count = serializers.ReadOnlyField()
    followers_count = serializers.ReadOnlyField()
    following_count = serializers.ReadOnlyField()

    def get_is_owner(self, obj):
        request = self.context['request']
```



models.py - be-inspired-drf-api

CI Python Linter

pep8ci.herokuapp.com/#

Profiles App – models.py

CI Python Linter

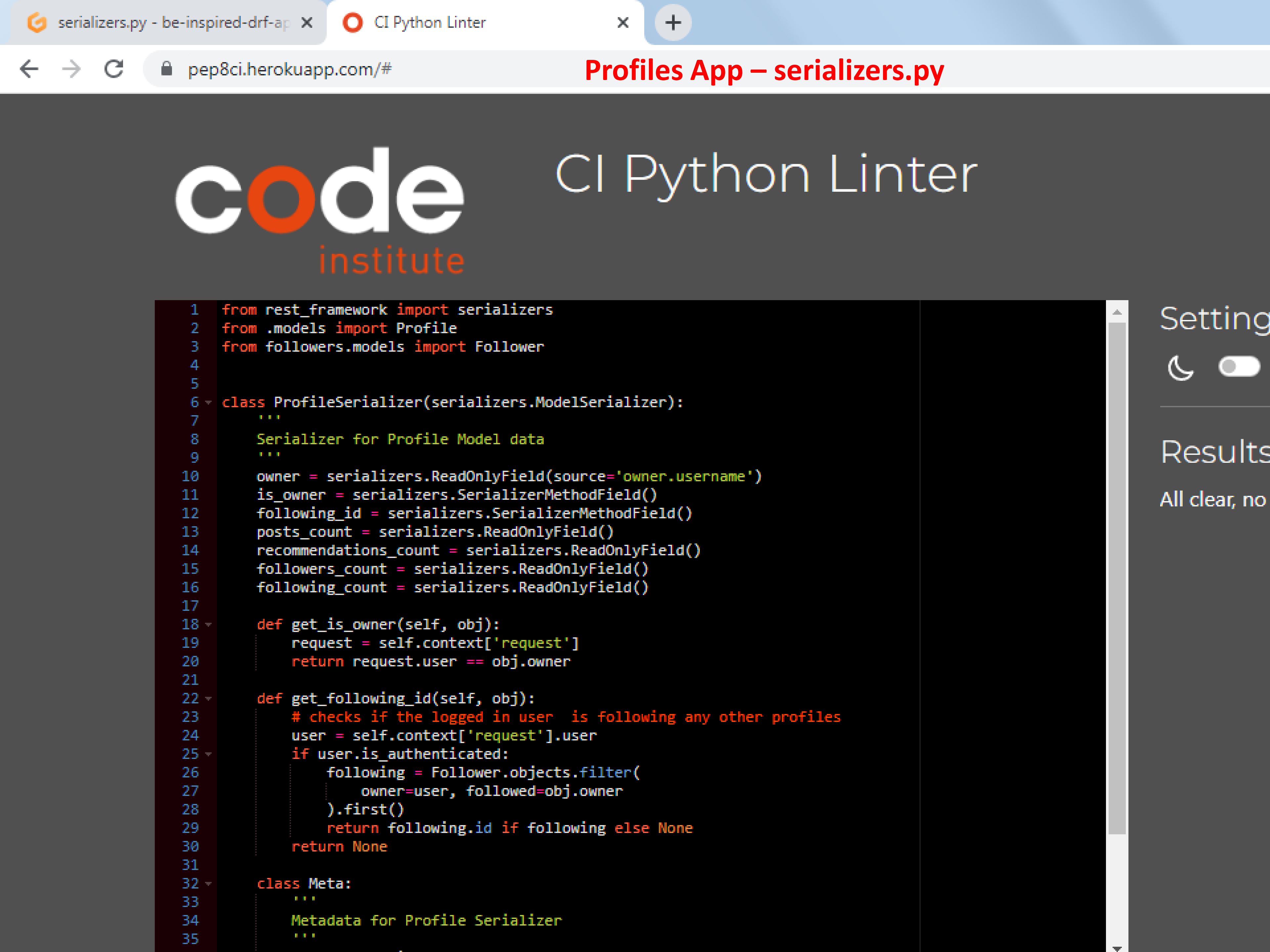
```python
from django.db import models
from django.db.models.signals import post_save
from django.contrib.auth.models import User


class Profile(models.Model):
    '''
    Profile model
    Extends user model
    '''
    class AgeGroup(models.TextChoices):
        """
        A class for the age_group key
        Contains different age ranges to choose from
        """
        TEENAGER = 'Teenager (10 - 18)',
        YOUNG_ADULT = 'Young Adult (19 - 25)',
        ADULT = 'Adult (26 - 40)',
        MIDDLE_AGED = 'Middle Aged (41 - 60)',
        SENIOR = 'Senior (>61)',

    owner = models.OneToOneField(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=200, blank=True)
    age_group = models.CharField(
        max_length=50,
        choices=AgeGroup.choices
        )
    brief_bio = models.TextField(blank=True)
    created_on = models.DateTimeField(auto_now_add=True)
    updated_on = models.DateTimeField(auto_now=True)
    # Default profile image
    image = models.ImageField(
        upload_to='images/', default='../default_profile_afxozd'
    )

```

Setting

Results

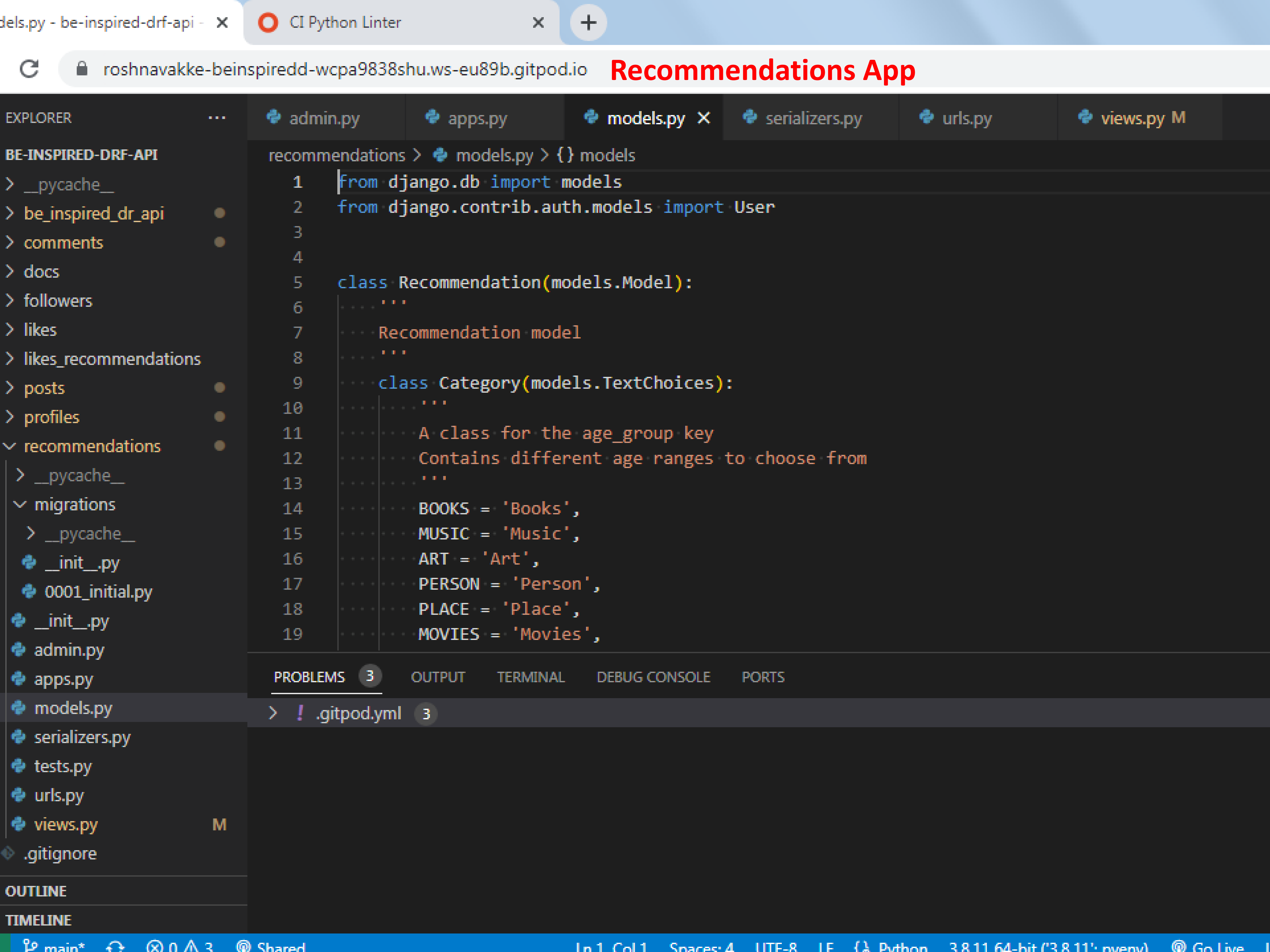
All clear, no

Profiles App – serializers.py



All clear, no

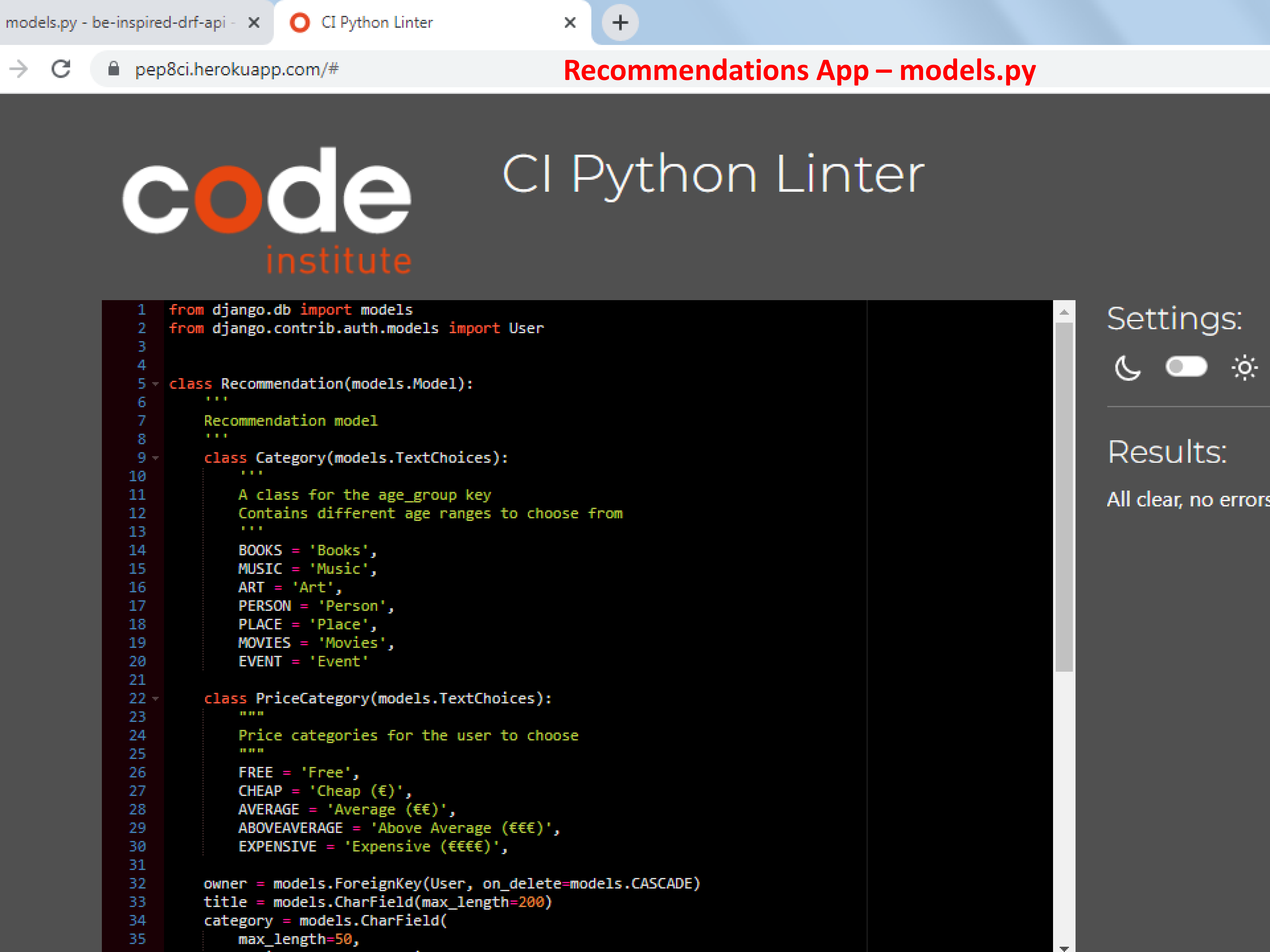
```
from rest_framework import serializers
from .models import Profile
from followers.models import Follower


class ProfileSerializer(serializers.ModelSerializer):
    '''
    Serializer for Profile Model data
    '''
    owner = serializers.ReadOnlyField(source='owner.username')
    is_owner = serializers.SerializerMethodField()
    following_id = serializers.SerializerMethodField()
    posts_count = serializers.ReadOnlyField()
    recommendations_count = serializers.ReadOnlyField()
    followers_count = serializers.ReadOnlyField()
    following_count = serializers.ReadOnlyField()

    def get_is_owner(self, obj):
        request = self.context['request']
        return request.user == obj.owner

    def get_following_id(self, obj):
        # checks if the logged in user  is following any other profiles
        user = self.context['request'].user
        if user.is_authenticated:
            following = Follower.objects.filter(
                owner=user, followed=obj.owner
            ).first()
            return following.id if following else None
        return None

    class Meta:
        '''
        Metadata for Profile Serializer
        '''
```



Recommendations App

```python
from django.db import models
from django.contrib.auth.models import User


class Recommendation(models.Model):
    '''
    Recommendation model
    '''
    class Category(models.TextChoices):
        '''
        A class for the age_group key
        Contains different age ranges to choose from
        '''
        BOOKS = 'Books',
        MUSIC = 'Music',
        ART = 'Art',
        PERSON = 'Person',
        PLACE = 'Place',
        MOVIES = 'Movies',
```

Recommendations App – models.py



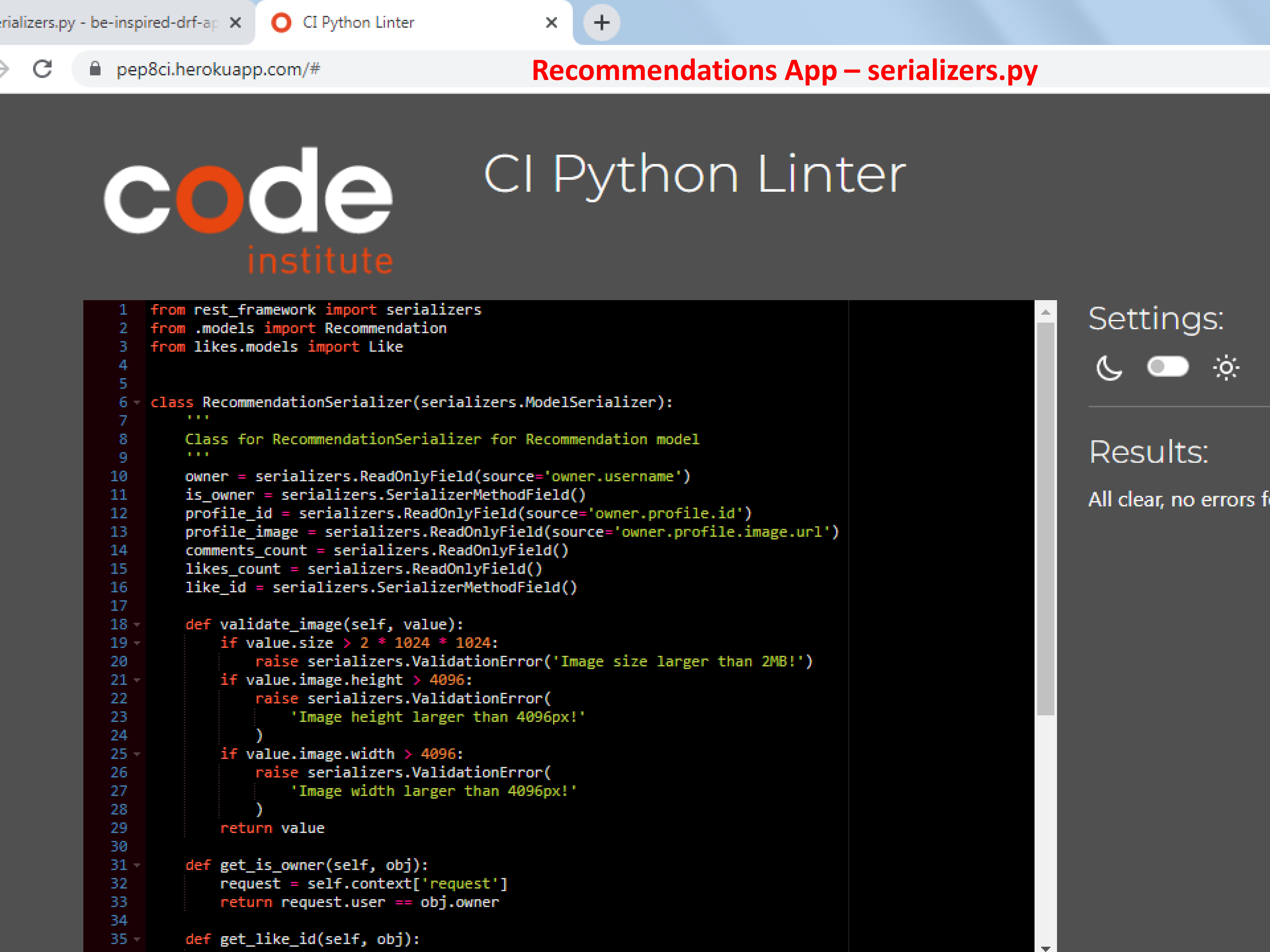
```
from django.db import models
from django.contrib.auth.models import User


class Recommendation(models.Model):
    '''
    Recommendation model
    '''
    class Category(models.TextChoices):
        '''
        A class for the age_group key
        Contains different age ranges to choose from
        '''
        BOOKS = 'Books',
        MUSIC = 'Music',
        ART = 'Art',
        PERSON = 'Person',
        PLACE = 'Place',
        MOVIES = 'Movies',
        EVENT = 'Event'

    class PriceCategory(models.TextChoices):
        """
        Price categories for the user to choose
        """
        FREE = 'Free',
        CHEAP = 'Cheap (€)',
        AVERAGE = 'Average (€€)',
        ABOVEAVERAGE = 'Above Average (€€€)',
        EXPENSIVE = 'Expensive (€€€€)',

    owner = models.ForeignKey(User, on_delete=models.CASCADE)
    title = models.CharField(max_length=200)
    category = models.CharField(
        max_length=50,
```

Settings:

Results:

All clear, no errors

Recommendations App – serializers.py



CI Python Linter

```
from rest_framework import serializers
from .models import Recommendation
from likes.models import Like


class RecommendationSerializer(serializers.ModelSerializer):
    '''
    Class for RecommendationSerializer for Recommendation model
    '''
    owner = serializers.ReadOnlyField(source='owner.username')
    is_owner = serializers.SerializerMethodField()
    profile_id = serializers.ReadOnlyField(source='owner.profile.id')
    profile_image = serializers.ReadOnlyField(source='owner.profile.image.url')
    comments_count = serializers.ReadOnlyField()
    likes_count = serializers.ReadOnlyField()
    like_id = serializers.SerializerMethodField()

    def validate_image(self, value):
        if value.size > 2 * 1024 * 1024:
            raise serializers.ValidationError('Image size larger than 2MB!')
        if value.image.height > 4096:
            raise serializers.ValidationError(
                'Image height larger than 4096px!'
            )
        if value.image.width > 4096:
            raise serializers.ValidationError(
                'Image width larger than 4096px!'
            )
        return value

    def get_is_owner(self, obj):
        request = self.context['request']
        return request.user == obj.owner

    def get_like_id(self, obj):
```

Settings:

Results:

All clear, no errors f