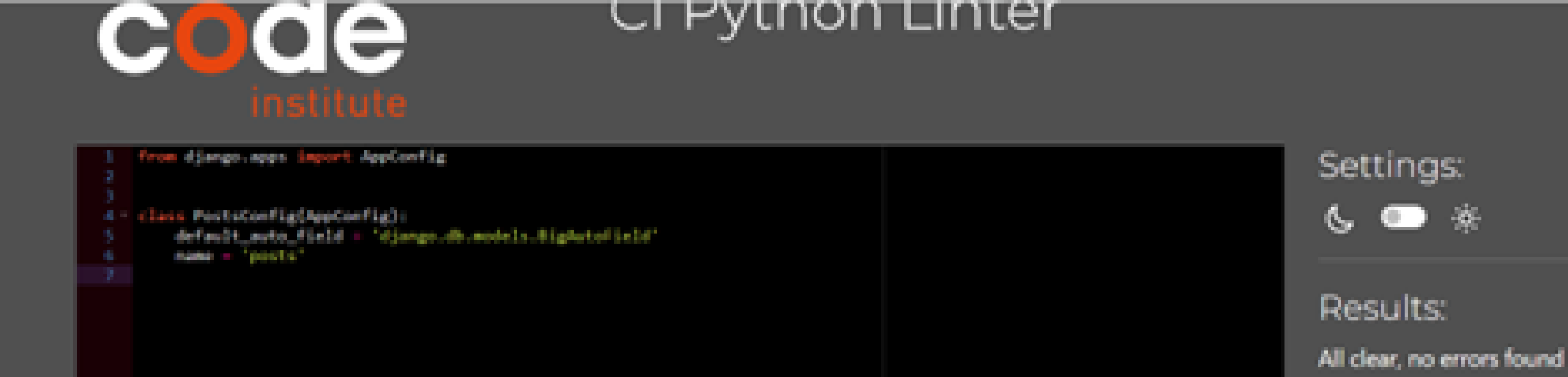
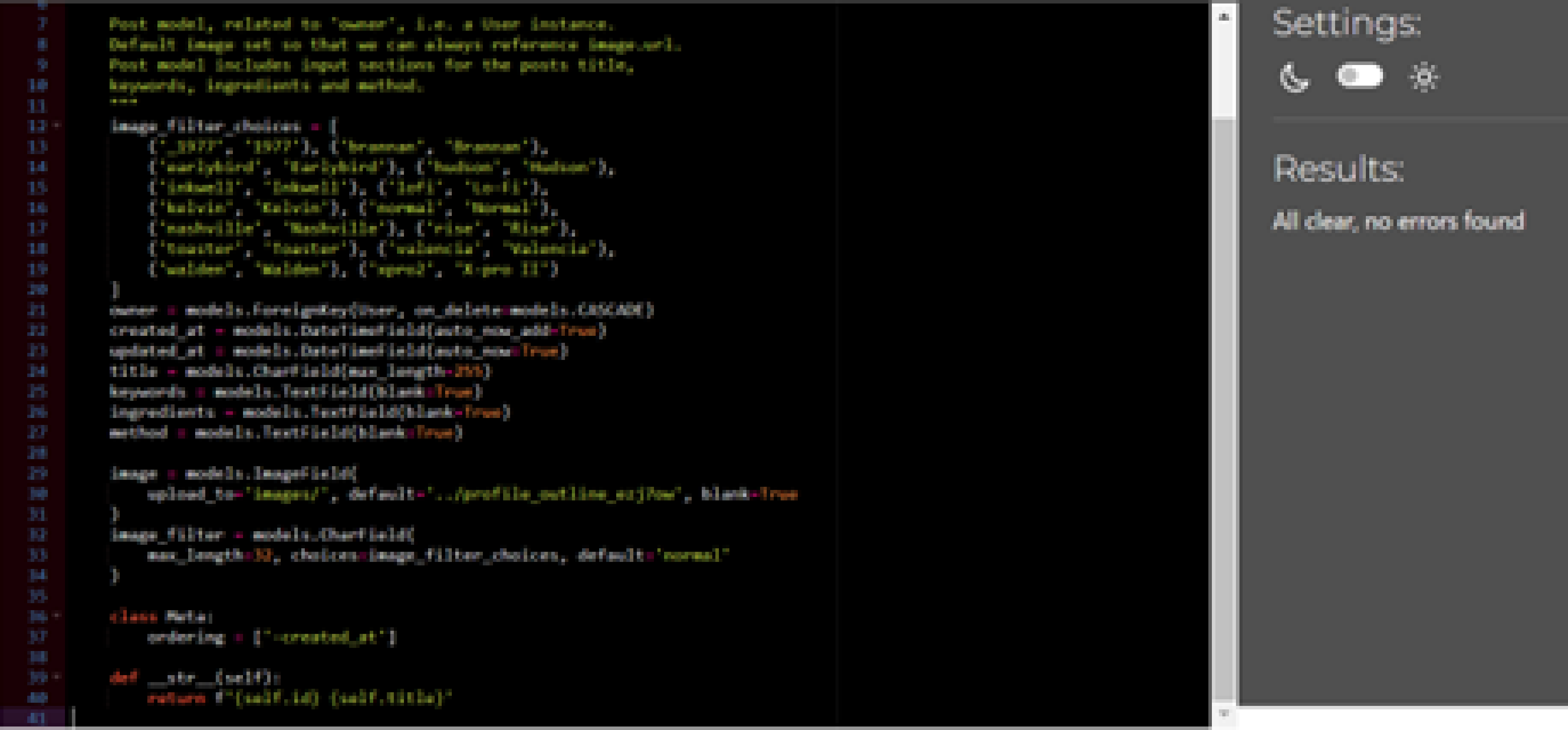


```
from django.apps import AppConfig


class PostsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'posts'
```

Settings:

Results:

All clear, no errors found

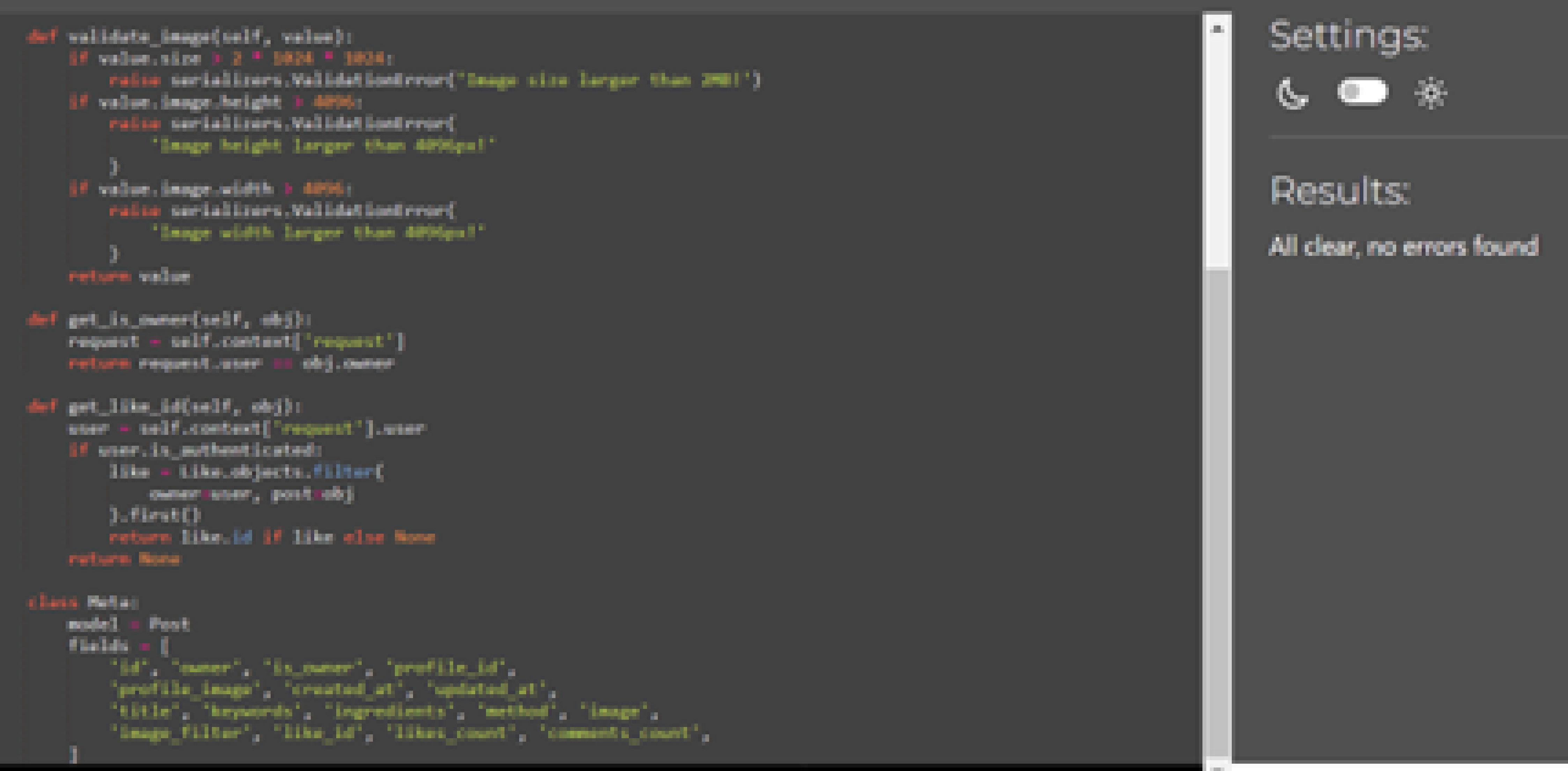


```
    Post model, related to 'owner', i.e. a User instance.
    Default image set so that we can always reference image.url.
    Post model includes input sections for the posts title,
    keywords, ingredients and method.
    """
    image_filter_choices = [
        ('_1977', '1977'), ('brannan', 'Brannan'),
        ('earlybird', 'Earlybird'), ('hudson', 'Hudson'),
        ('inkwell', 'Inkwell'), ('lofi', 'Lo-Fi'),
        ('kelvin', 'Kelvin'), ('normal', 'Normal'),
        ('nashville', 'Nashville'), ('rise', 'Rise'),
        ('toaster', 'Toaster'), ('valencia', 'Valencia'),
        ('walden', 'Walden'), ('xpro2', 'X-pro II')
    ]
    owner = models.ForeignKey(User, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    title = models.CharField(max_length=255)
    keywords = models.TextField(blank=True)
    ingredients = models.TextField(blank=True)
    method = models.TextField(blank=True)

    image = models.ImageField(
        upload_to='images/', default='../profile_outline_ezj7ow', blank=True
    )
    image_filter = models.CharField(
        max_length=32, choices=image_filter_choices, default='normal'
    )

    class Meta:
        ordering = ['-created_at']

    def __str__(self):
        return f'{self.id} {self.title}'
```

Settings:

Results:

All clear, no errors found

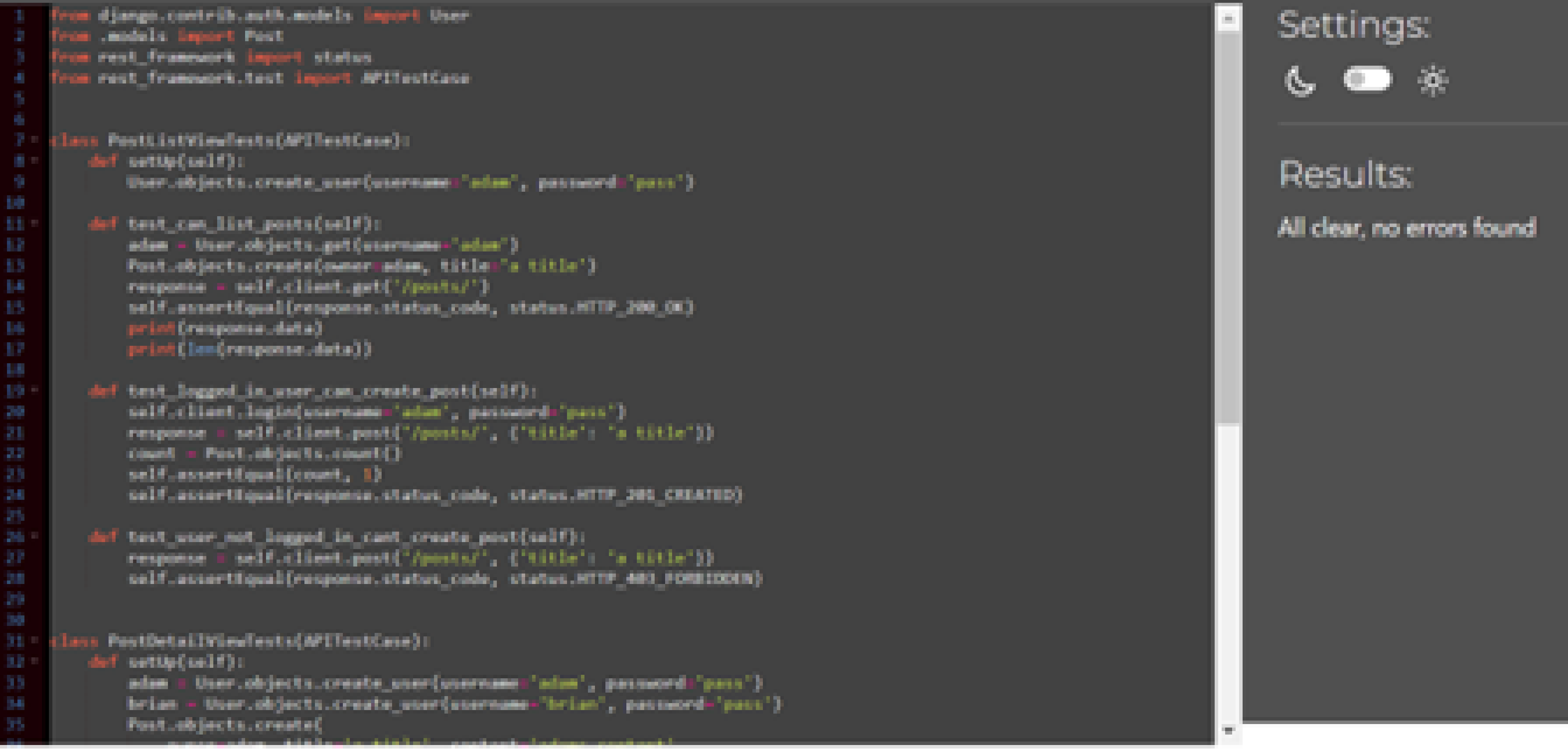


```
def validate_image(self, value):
    if value.size > 2 * 1024 * 1024:
        raise serializers.ValidationError('Image size larger than 2MB!')
    if value.image.height > 4096:
        raise serializers.ValidationError(
            'Image height larger than 4096px!'
        )
    if value.image.width > 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):
    user = self.context['request'].user
    if user.is_authenticated:
        like = Like.objects.filter(
            owner=user, post=obj
        ).first()
        return like.id if like else None
    return None

class Meta:
    model = Post
    fields = [
        'id', 'owner', 'is_owner', 'profile_id',
        'profile_image', 'created_at', 'updated_at',
        'title', 'keywords', 'ingredients', 'method', 'image',
        'image_filter', 'like_id', 'likes_count', 'comments_count',
    ]
```

Settings:

Results:

All clear, no errors found

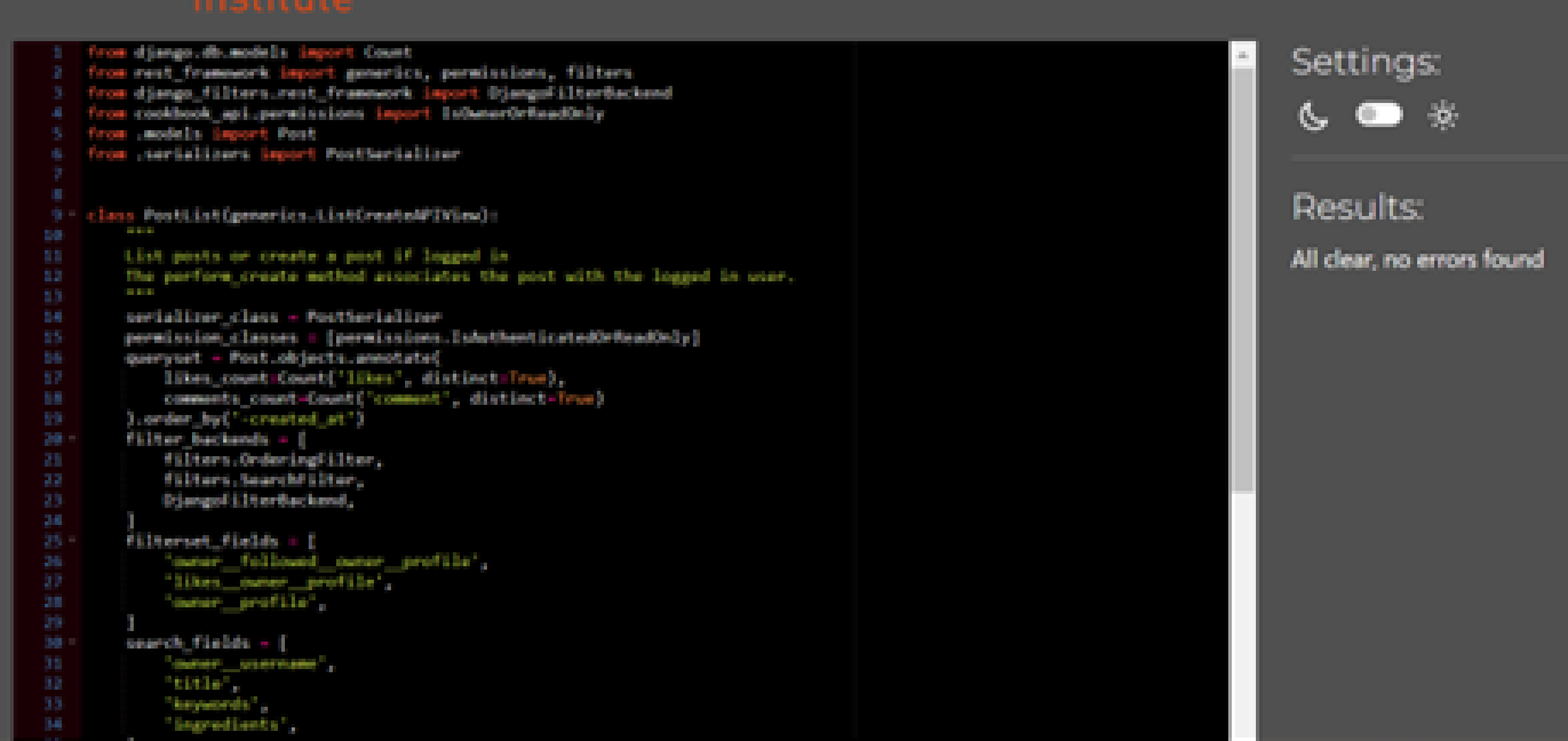


```python
from django.contrib.auth.models import User
from .models import Post
from rest_framework import status
from rest_framework.test import APITestCase


class PostListViewTests(APITestCase):
    def setUp(self):
        User.objects.create_user(username='adam', password='pass')

    def test_can_list_posts(self):
        adam = User.objects.get(username='adam')
        Post.objects.create(owner=adam, title='a title')
        response = self.client.get('/posts/')
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        print(response.data)
        print(len(response.data))

    def test_logged_in_user_can_create_post(self):
        self.client.login(username='adam', password='pass')
        response = self.client.post('/posts/', {'title': 'a title'})
        count = Post.objects.count()
        self.assertEqual(count, 1)
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)

    def test_user_not_logged_in_cant_create_post(self):
        response = self.client.post('/posts/', {'title': 'a title'})
        self.assertEqual(response.status_code, status.HTTP_403_FORBIDDEN)


class PostDetailViewTests(APITestCase):
    def setUp(self):
        adam = User.objects.create_user(username='adam', password='pass')
        brian = User.objects.create_user(username='brian', password='pass')
        Post.objects.create(
            owner=adam, title='a title', content='adams content'
```

Settings:

Results:

All clear, no errors found



```python
from django.db.models import Count
from rest_framework import generics, permissions, filters
from django_filters.rest_framework import DjangoFilterBackend
from cookbook_api.permissions import IsOwnerOrReadOnly
from .models import Post
from .serializers import PostSerializer


class PostList(generics.ListCreateAPIView):
    """
    List posts or create a post if logged in
    The perform_create method associates the post with the logged in user.
    """
    serializer_class = PostSerializer
    permission_classes = [permissions.IsAuthenticatedOrReadOnly]
    queryset = Post.objects.annotate(
        likes_count=Count('likes', distinct=True),
        comments_count=Count('comment', distinct=True)
    ).order_by('-created_at')
    filter_backends = [
        filters.OrderingFilter,
        filters.SearchFilter,
        DjangoFilterBackend,
    ]
    filterset_fields = [
        'owner__followed__owner__profile',
        'likes__owner__profile',
        'owner__profile',
    ]
    search_fields = [
        'owner__username',
        'title',
        'keywords',
        'ingredients',
```

Results:

All clear, no errors found