

# Nomad Narratives

## PEP8 Validation

The following is a series of screenshots from the CI Python Linter broken down by relevant apps. Flake8 and Pylint were used throughout development to highlight PEP8 issues as they arose.

# Comments App

## admin.py - Passed

```
1 """
2 Django admin configuration for the Comment model.
3 """
4 from django.contrib import admin
5 from .models import Comment
6
7 admin.site.register(Comment)
8
```

Settings:



Results:

All clear, no errors found

# Comments App

## apps.py - Passed

```
1 """
2 App configuration for the Comments application.
3
4 This module defines the configuration settings for the Comments app
5 within the Django project.
6 """
7 from django.apps import AppConfig
8
9
10 class CommentsConfig(AppConfig):
11     """
12     This class sets the default primary key type and specifies the app name.
13     """
14     default_auto_field = 'django.db.models.BigAutoField'
15     name = 'comments'
16
```

Settings:



Results:

All clear, no errors found

# Comments App

## models.py - Passed

```
1 """
2     Represents a comment made by a user on a specific trip post.
3     ...
4
5     from django.db import models
6     from django.contrib.auth.models import User
7     from posts.models import TripPost
8
9
10    class Comment(models.Model):
11        ...
12            Comment model, related to User and Post.
13
14        ...
15            owner = models.ForeignKey(User, on_delete=models.CASCADE)
16            post = models.ForeignKey(TripPost, on_delete=models.CASCADE)
17            created_at = models.DateTimeField(auto_now_add=True)
18            updated_at = models.DateTimeField(auto_now=True)
19            content = models.TextField()
20
21    class Meta:
22        ...
23            Orders comments by creation time in descending order.
24        ...
25            ordering = ['-created_at']
26
27    def __str__(self):
28        ...
29            Returns a string representation of the comment's content.
30        ...
31            return str(self.content)
```

Settings:



Results:

All clear, no errors found

# Comments App

## serializers.py - Passed

```
1 from rest_framework import serializers
2 from django.contrib.humanize.templatetags.humanize import naturaltime
3 from .models import Comment
4
5
6 class CommentSerializer(serializers.ModelSerializer):
7     """
8         Serializer for Comment model.
9     """
10    owner = serializers.ReadOnlyField(source='owner.username')
11    is_owner = serializers.SerializerMethodField()
12    profile_id = serializers.ReadOnlyField(source='owner.profile.id')
13    profile_image = serializers.ReadOnlyField(source='owner.profile.image.url')
14    created_at = serializers.SerializerMethodField()
15    updated_at = serializers.SerializerMethodField()
16
17    def get_is_owner(self, obj):
18        """
19            Checks for owner of comment.
20        """
21        request = self.context['request']
22        return request.user == obj.owner
23
24    def get_created_at(self, obj):
25        """
26            Returns as human readable time.
27        """
28        return naturaltime(obj.created_at)
29
30    def get_updated_at(self, obj):
31        """
```

Settings:



Results:

All clear, no errors found

# Comments App

## tests.py

*Empty file -*

*Passed but no validation  
needed.*

The screenshot shows the VS Code interface. At the top, there's a tab bar with 'tests.py' selected. Below the tabs, the breadcrumb navigation shows 'nomad-narratives > comments > tests.py'. The main editor area is mostly empty, with a single number '1' at the top left. At the bottom of the screen is the 'PROBLEMS' panel, which has tabs for 'PROBLEMS', 'OUTPUT', and '...', with 'PROBLEMS' being the active tab. A search bar labeled 'Filter (e.g. text, \*\*/\*.ts, !\*\*/node\_modules/\*\*)' is located at the top of the panel. The message 'No problems have been detected in the workspace.' is displayed in the panel.

# Comments App

## urls.py - Passed

```
1 """
2 URL configuration for the Comments application.
3
4 This module defines the URL patterns for the Comments app, mapping
5 views to specific endpoints for handling comment-related requests.
6 """
7 from django.urls import path
8 from comments import views
9
10 urlpatterns = [
11     path('comments/', views.CommentList.as_view()),
12     path('comments/<int:pk>/', views.CommentDetail.as_view())
13 ]
14 ]
```

Settings:



Results:

All clear, no errors found

# Comments App

## views.py - Passed

```
1  """
2  Views for the Comments application.
3
4  This module defines API views for listing, creating, retrieving, updating,
5  and deleting comments using Django REST Framework.
6  """
7
8  from rest_framework import generics, permissions
9  from django_filters.rest_framework import DjangoFilterBackend
10 from nomadnarrativesapi.permissions import IsOwnerOrReadOnly
11 from .models import Comment
12 from .serializers import CommentSerializer, CommentDetailSerializer
13
14 class CommentList(generics.ListCreateAPIView):
15     """
16         List all comments
17         Create a new comment if authenticated
18         Associate the current logged in user with the comment
19     """
20
21     permission_classes = [permissions.IsAuthenticatedOrReadOnly]
22     serializer_class = CommentSerializer
23     queryset = Comment.objects.all()
24
25     filter_backends = [
26         DjangoFilterBackend
27     ]
28     filterset_fields = [
29         'post'
30     ]
```

Settings:



Results:

All clear, no errors found

# Contact App

## admin.py - Passed

```
1 """
2 Admin configuration for the Contact model.
3 """
4 from django.contrib import admin
5 from .models import Contact
6
7
8 @admin.register(Contact)
9 class ContactAdmin(admin.ModelAdmin):
10     """
11     Admin interface for managing Contact model entries.
12
13     - Displays sender name, email, and creation date in the admin list view.
14     """
15     list_display = ('sender_name', 'email', 'created_at')
16     search_fields = ('sender_name', 'last_name', 'email')
17
```

Settings:



Results:

All clear, no errors found

# Contact App

## apps.py - Passed

```
1 """
2 App configuration for the Contact application.
3 """
4
5 from django.apps import AppConfig
6
7
8 class ContactConfig(AppConfig):
9     """
10     Configuration class for the Contact app.
11     Sets the default auto field type and specifies the app's name.
12     """
13     default_auto_field = 'django.db.models.BigAutoField'
14     name = 'contact'
15 |
```

Settings:



Results:

All clear, no errors found

# Contact App

## models.py - Passed

```
1 """
2 Defines the Contact model for storing user-submitted contact form data.
3 """
4 from django.db import models
5
6
7 class Contact(models.Model):
8     """
9         Fields for form to contact admin.
10    """
11    sender_name = models.CharField(max_length=100)
12    email = models.EmailField()
13    message = models.TextField()
14    created_at = models.DateTimeField(auto_now_add=True)
15
16    def __str__(self):
17        return f"{self.sender_name}"
18
```

Settings:



Results:

All clear, no errors found

# Contact App

## serializers.py - Passed

```
1 """
2 Serializer for the Contact model.
3 """
4
5 from rest_framework import serializers
6 from .models import Contact
7
8
9 class ContactSerializer(serializers.ModelSerializer):
10     """
11     Serializer for contact model.
12     """
13     class Meta:
14         """
15         Returns all fields.
16         """
17         model = Contact
18         fields = '__all__'
19
```

Settings:

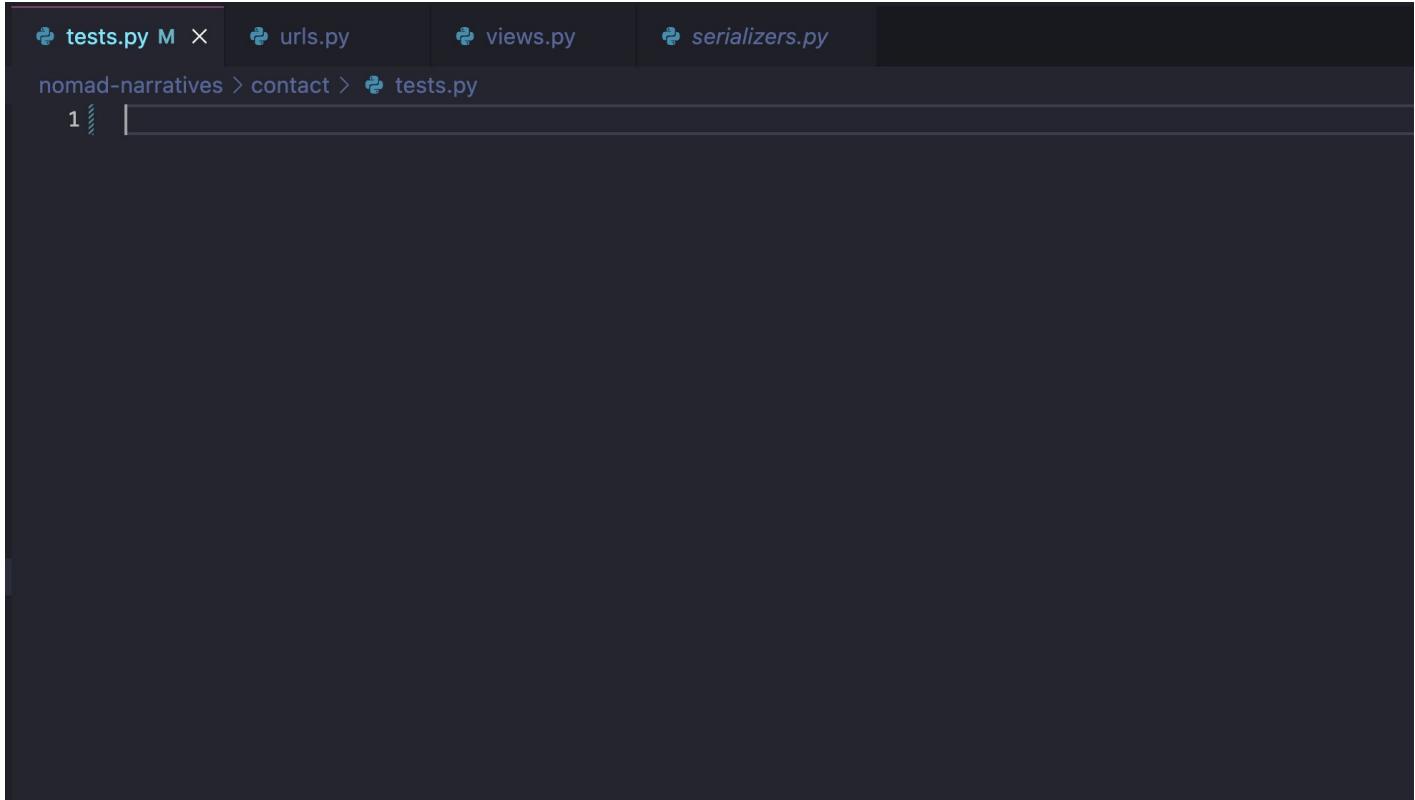


Results:

All clear, no errors found

# Contact App

**tests.py - Passed**



A screenshot of a code editor window titled "tests.py M X". The window shows the file structure: "nomad-narratives > contact > tests.py". The code editor interface includes tabs for urls.py, views.py, and serializers.py. The main code area is dark, indicating it is empty.

Empty file

# Contact App

## urls.py - Passed

```
1 """
2 URL configuration for the Contact app.
3
4 Defines the URL pattern for the contact form API endpoint.
5 """
6
7 from django.urls import path
8 from contact import views
9
10 urlpatterns = [
11     path('contact/', views.ContactCreateView.as_view(), name='contact-create'),
12 ]
13
```

Settings:



Results:

All clear, no errors found

# Contact App

## views.py - Passed

```
1 """
2 Defines the ContactCreateView, which provides an API endpoint
3 for creating Contact model instances.
4 """
5 from rest_framework import generics
6 from .models import Contact
7 from .serializers import ContactSerializer
8
9
10 class ContactCreateView(generics.CreateAPIView):
11     """
12     View for contact model.
13     """
14     queryset = Contact.objects.all() # pylint: disable=no-member
15     serializer_class = ContactSerializer
16 |
```

Settings:



Results:

All clear, no errors found

# Followers App

## admin.py - Passed

```
1 """
2 Django admin configuration for the Follower model.
3 """
4 from django.contrib import admin
5 from .models import Follower
6
7 admin.site.register(Follower)
8
```

Settings:



Results:

All clear, no errors found

# Followers App

## apps.py - Passed

```
1 """
2 App configuration for the Followers application.
3
4 This module defines the configuration settings for the Followers app
5 within the Django project.
6 """
7 from django.apps import AppConfig
8
9
10 class FollowersConfig(AppConfig):
11     """
12         Configuration class for the Followers app.
13
14     This class sets the default primary key type and specifies the app name.
15     """
16     default_auto_field = 'django.db.models.BigAutoField'
17     name = 'followers'
```

Settings:



Results:

All clear, no errors found

# Followers App

## models.py - Passed

```
1  """
2  The `Follower` model tracks which users are following other users.
3  """
4
5  from django.db import models
6  from django.contrib.auth.models import User
7
8
9  class Follower(models.Model):
10     """
11     Represents a follower relationship between users.
12
13     - 'owner' refers to the user who is following another user.
14     - 'followed' refers to the user being followed.
15     - The 'related_name' attribute distinguishes the two User instances.
16     - The 'unique_together' constraint ensures a user cannot
17     follow the same user more than once.
18     """
19     owner = models.ForeignKey(
20         User,
21         on_delete=models.CASCADE,
22         related_name='following'
23     )
24     followed = models.ForeignKey(
25         User,
26         on_delete=models.CASCADE,
27         related_name='followed'
28     )
29     created_at = models.DateTimeField(
30         auto_now_add=True
31     )
```

Settings:



Results:

All clear, no errors found

# Followers App

## serializers.py - Passed

```
1 """
2     Serializer for the `Follower` model, handling the creation and serialization
3     of follower relationships between users.
4
5     Includes validation for duplicate follow attempts.
6 """
7 from django.db import IntegrityError
8 from rest_framework import serializers
9 from .models import Follower
10
11
12 class FollowerSerializer(serializers.ModelSerializer):
13     """
14         Serializer for Followers model.
15
16         It handles serialization of the `owner` and `followed`
17         user details, as well as managing errors related to duplicate
18         follow requests.
19     """
20     owner = serializers.ReadOnlyField(source='owner.username')
21     followed_name = serializers.ReadOnlyField(source='followed.username')
22
23 class Meta:
24     """
25         Specifies what fields to be returned.
26     """
27     model = Follower
28     fields = ['id', 'owner', 'created_at', 'followed', 'followed_name']
29
30 def create(self, validated_data):
31     """
```

Settings:



Results:

All clear, no errors found

# Followers App

## tests.py

*Empty file -*

*Passed but no validation  
needed.*

The screenshot shows a dark-themed code editor interface. At the top, there's a tab bar with 'tests.py' and an 'X' button. Below it, a breadcrumb navigation bar shows 'nomad-narratives > followers > tests.py'. A status bar at the bottom indicates '1' file. The main workspace is empty. At the bottom, there's a 'Problems' panel with a tab bar labeled 'PROBLEMS' (which is underlined), 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and '...'. To the right of the tabs is a 'Filter (e.g. text, \*\*/\*.ts, !\*\*/no)' input field. The 'PROBLEMS' tab displays the message: 'No problems have been detected in the workspace.'

# Followers App

## urls.py - Passed

```
1 """
2 Defines URL patterns for managing follower relationships,
3 including listing followers and viewing details of a specific
4 follower relationship.
5 """
6
7 from django.urls import path
8 from followers import views
9
10 urlpatterns = [
11     path('followers/', views.FollowerList.as_view()),
12     path('followers/<int:pk>/', views.FollowerDetail.as_view())
13 ]
14 |
```

Settings:



Results:

All clear, no errors found

# Followers App

## views.py - Passed

```
13 from rest_framework import generics, permissions
14 from nomadnarrativesapi.permissions import IsOwnerOrReadOnly
15 from .models import Follower
16 from .serializers import FollowerSerializer
17
18
19 - class FollowerList(generics.ListCreateAPIView):
20     """
21         API view to list all followers or create a new follower relationship.
22
23         Allows authenticated users to create a new follow relationship,
24         automatically setting the authenticated user as the owner of the follow.
25         Read-only access is available to non-authenticated users.
26     """
27     serializer_class = FollowerSerializer
28     queryset = Follower.objects.all()
29     permission_classes = [permissions.IsAuthenticatedOrReadOnly]
30
31 -     def perform_create(self, serializer):
32         serializer.save(owner=self.request.user)
33
34
35 - class FollowerDetail(generics.RetrieveDestroyAPIView):
36     """
37         Retrieves a follower.
38         Allows users to follow or unfollow another user.
39     """
40     serializer_class = FollowerSerializer
41     queryset = Follower.objects.all()
42     permission_classes = [IsOwnerOrReadOnly]
43
```

Settings:



Results:

All clear, no errors found

# Likes App

## admin.py - Passed

```
1 """
2 Django admin configuration for the Like model.
3 """
4 from django.contrib import admin
5 from .models import Like
6
7 admin.site.register(Like)
8 |
```

Settings:



Results:

All clear, no errors found

# Likes App

## apps.py - Passed

```
1 """
2 Configures the "likes" app, defining the default auto field type and app name
3 .
4 from django.apps import AppConfig
5
6
7 class LikesConfig(AppConfig):
8     """
9         Configuration for the "likes" app, specifying the default auto field type
10        and the app name.
11        """
12    default_auto_field = 'django.db.models.BigAutoField'
13    name = 'likes'
14
```

Settings:



Results:

All clear, no errors found

# Likes App

## models.py - Passed

```
1 """
2 Contains the `Like` model, representing the relationship between users and
3 posts they have liked. Ensures that a user can like a post only once.
4 """
5 from django.db import models
6 from django.contrib.auth.models import User
7 from posts.models import TripPost
8
9
10 class Like(models.Model):
11     """
12         Like model, related to Owner and TripPost.
13         'unique_together' ensures a user can't like one post twice.
14     """
15     owner = models.ForeignKey(User, on_delete=models.CASCADE)
16     post = models.ForeignKey(
17         TripPost,
18         on_delete=models.CASCADE,
19         related_name='likes'
20     )
21     created_at = models.DateTimeField(auto_now_add=True)
22
23     class Meta:
24         """
25             Represents a like on a post by a user.
26
27             Ensures that each user can like a post only once by enforcing a
28             unique constraint between the user and post.
29         """
30         ordering = ['-created_at']
31         unique_together = ['owner', 'post']
```

Settings:



Results:

All clear, no errors found

# Likes App

## serializers.py - Passed

```
1  """
2 Contains the serializer for the `Like` model, which handles the creation
3 and serialization of "like" relationships between users and posts.
4 Includes error handling for duplicate likes.
5 """
6 from django.db import IntegrityError
7 from rest_framework import serializers
8 from .models import Like
9
10
11 class LikeSerializer(serializers.ModelSerializer):
12     """
13     Serializer for Like model.
14     """
15     owner = serializers.ReadOnlyField(source='owner.username')
16
17     class Meta:
18         """
19         Specifies what fields to be returned.
20         """
21         model = Like
22         fields = ['id', 'created_at', 'owner', 'post']
```

Settings:



Results:

All clear, no errors found

# Likes App

## tests.py

*Empty file -*

*Passed but no validation  
needed.*

The screenshot shows a dark-themed code editor interface. At the top, there are two tabs: "serializers.py" and "tests.py". The "tests.py" tab is active, showing the path "nomad-narratives > likes > tests.py". Below the tabs, the code editor area is empty, with a single digit "1" indicating the current line. At the bottom of the screen, there is a navigation bar with several tabs: "PROBLEMS" (which is underlined in red), "OUTPUT", "DEBUG CONSOLE", "TERMINAL", "...", and a search bar labeled "Filter (e.g. ...)". The main message in the "PROBLEMS" tab is "No problems have been detected in the workspace."

# Likes App

## urls.py - Passed

```
1 """
2 Defines URL patterns for managing "like" relationships, including listing
3     likes
4 and viewing details of a specific like.
5 """
6 from django.urls import path
7 from likes import views
8
9 urlpatterns = [
10     path('likes/', views.LikeList.as_view()),
11     path('likes/<int:pk>/', views.LikeDetail.as_view()),
12 ]
```

Settings:



Results:

All clear, no errors found

# Likes App

## views.py - Passed

```
1 """
2 Contains API views for managing "like" relationships between users and
3     posts.
4
5 - `LikeList`: Lists all likes and allows authenticated users to create
6     a new like for a post. Non-authenticated users have read-only access.
7
8 - `LikeDetail`: Retrieves or deletes a specific like. Only the owner of the
9     like can modify or remove it.
10 """
11 from rest_framework import generics, permissions
12 from nomadnarrativesapi.permissions import IsOwnerOrReadOnly
13 from .models import Like
14 from .serializers import LikeSerializer
15
16 class LikeList(generics.ListCreateAPIView):
17     """
18     Lists all likes.
19     Creates a like if user is authenticated.
20     The perform_create method links the like with the logged in user.
21     """
```

Settings:



Results:

All clear, no errors found

# Posts App

## admin.py - Passed

```
1 """
2 Admin configuration for the TripPost and TripDetails models.
3 - Manages trip-related data, including inline editing of TripDetails
4 within TripPost.
5 """
6 from django.contrib import admin
7 from .models import TripPost, TripDetails
8
9
10 class TripDetailsInline(admin.StackedInline):
11     """
12         TripDetails Inline Admin: Allow editing TripDetails
13         directly within TripPost.
14     """
15     model = TripDetails
16     extra = 1 # Number of empty forms to show by default
17     fields = (
18         'country', 'city', 'traveller_number', 'relevant_for',
19         'duration_value', 'duration_unit'
20     ) # Fields to display in the inline form
21
22     def get_cities(self, obj):
23         """
24             Custom method to display a list of cities as a
25             comma-separated string.
26         """
```

Settings:



Results:

All clear, no errors found

# Posts App

## apps.py - Passed

```
1 """
2 App configuration for the Posts application.
3
4 This module defines the configuration settings for the Posts app
5 within the Django project.
6 """
7 from django.apps import AppConfig
8
9
10 class PostsConfig(AppConfig):
11     """
12         This class sets the default primary key type and specifies the app name.
13     """
14     default_auto_field = 'django.db.models.BigAutoField'
15     name = 'posts'
16 |
```

Settings:



Results:

All clear, no errors found

# Posts App

## models.py - Passed

```
1  from django.db import models
2  from django.core.exceptions import ValidationError
3  from django.contrib.auth.models import User
4  from cities_light.models import City, Country
5  from utils_continents import get_continent_by_country
6
7
8
9  class TripPost(models.Model):
10     """
11         Trip Post model, related to 'owner', i.e. a User instance.
12         Default image set so that we can always reference image.url.
13
14         This model handles the content of the post (image, article content, title).
15         This model does not handle the details of the trip (see TripDetails model).
16     """
17
18     owner = models.ForeignKey(
19         User,
20         on_delete=models.CASCADE
21     )
22     created_at = models.DateTimeField(
23         auto_now_add=True
```

Settings:



Results:

All clear, no errors found

# Posts App

## serializers.py - Passed

```
1 """
2 Posts Serializer
3 """
4 from rest_framework import serializers
5 from cities_light.models import City, Country
6 from likes.models import Like
7 from .models import TripPost, TripDetails
8
9
10 class TripDetailsSerializer(serializers.ModelSerializer):
11     """
12     Serializer for TripDetails model.
13     """
14     continent = serializers.ReadOnlyField()
15     country = serializers.PrimaryKeyRelatedField(
16         queryset=Country.objects.all())
17     city = serializers.PrimaryKeyRelatedField(
18         queryset=City.objects.all()
19     )
```

Settings:



Results:

All clear, no errors found

# Posts App

## tests.py

*Empty file -*

*Passed but no validation  
needed.*

A screenshot of a code editor interface, likely VS Code, showing an empty Python test file named 'tests.py'. The file path is 'nomad-narratives > posts > tests.py'. The code editor has a dark background. At the bottom, there is a navigation bar with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and Filter (e.g.). A message in the PROBLEMS tab states: 'No problems have been detected in the workspace.'

```
tests.py
nomad-narratives > posts > tests.py
1
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    ...    Filter (e.g.)

No problems have been detected in the workspace.

# Posts App

## urls.py - Passed

```
1 """
2 URL configuration for the posts app.
3
4 This module defines the URL patterns for the post-related views, including:
5
6 - A list view for all posts (`PostList`).
7 - A detail view for individual posts (`PostDetail`).
8 """
9 from django.urls import path
10 from posts import views
11
12 urlpatterns = [
13     path('posts/', views.PostList.as_view()),
14     path('posts/<int:pk>/', views.PostDetail.as_view()),
15 ]
16
```

### Settings:



### Results:

All clear, no errors found

# Posts App

## views.py - Passed

```
1 """
2 Imports for views related to TripPost.
3
4 This module imports various libraries and components used for handling
5 TripPost views, including Django, REST framework components, and custom
6 permissions and serializers.
7 """
8 from django.db.models import Count
9 from django.shortcuts import get_object_or_404
10 from rest_framework import generics, permissions, filters
11 from django_filters.rest_framework import DjangoFilterBackend
12 from nomadnarrativesapi.permissions import IsOwnerOrReadOnly
13 from utils.continents import get_continent_by_country
14 from cities_light.models import Country, City
15 from .serializers import TripPostSerializer
16 from .models import TripPost
17
18
19 class PostList(generics.ListCreateAPIView):
20     """
21         Handles Post logic.
22     """
```

Settings:



Results:

All clear, no errors found

# Profiles App

## admin.py - Passed

```
1 """
2 Django admin configuration for the Profile model.
3 """
4 from django.contrib import admin
5 from .models import Profile
6
7 admin.site.register(Profile)
8
```

Settings:



Results:

All clear, no errors found

# Profiles App

## apps.py - Passed

```
1 """
2 Configures the "profiles" app, specifying the default auto field type
3 and the app name.
4 """
5 from django.apps import AppConfig
6
7
8 class ProfilesConfig(AppConfig):
9     """
10         This class specifies the default auto field type for the app
11         (BigAutoField) and sets the name of the app to "Profiles".
12     """
13     default_auto_field = 'django.db.models.BigAutoField'
14     name = 'profiles'
15
```

Settings:



Results:

All clear, no errors found

# Profiles App

## models.py - Passed

```
1 """
2 Defines the Profile model and its related logic for user profiles.
3
4 - `Profile`: Stores user profile information such as name, content (bio),
5   and image.
6 - A new profile is automatically created when a new User instance is created.
7 """
8
9 from django.db import models
10 from django.db.models.signals import post_save
11 from django.dispatch import receiver
12 from django.contrib.auth.models import User
13
14
15 class Profile(models.Model):
16     """
17     Profile model.
18     """
19     owner = models.OneToOneField(User, on_delete=models.CASCADE)
20     created_at = models.DateTimeField(auto_now_add=True)
21     updated_at = models.DateTimeField(auto_now=True)
22     name = models.CharField(max_length=255, blank=True)
23     content = models.TextField(blank=True)
24     image = models.ImageField(
25         upload_to='images/', default='../default_profile_cf7kd5'
26     )
```

Settings:



Results:

All clear, no errors found

# Profiles App

## serializers.py - Passed

```
1 """
2     Serializer for the Profile model and related fields.
3
4     - `ProfileSerializer`: Serializes Profile model data, including user
5         profile
6         details and dynamic fields like follow status, post counts, and
7         follower counts.
8
9     ...
10    from rest_framework import serializers
11    from followers.models import Follower
12    from .models import Profile
13
14
15    class ProfileSerializer(serializers.ModelSerializer):
16        """
17            Serializer for Profile model.
```

Settings:



Results:

All clear, no errors found

# Profiles App

## tests.py

*Empty file -*

*Passed but no validation  
needed.*

The screenshot shows a dark-themed code editor interface. At the top, a tab bar displays the file name "tests.py" with a "M" icon and a close button. Below the tab bar, the breadcrumb navigation shows the path: "nomad-narratives > profiles > tests.py". The main workspace area is empty, with a cursor at position 1. At the bottom of the editor, there is a status bar with tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and an ellipsis "...". To the right of the tabs is a search input field labeled "Filter (e.g. ...)". Below the tabs, a message states "No problems have been detected in the workspace."

# Profiles App

## urls.py - Passed

```
1 """
2 Defines URL routes for accessing and interacting with user profiles.
3
4 - `ProfileList`: View for listing all profiles and creating new profiles.
5 - `ProfileDetail`: View for retrieving, updating, and
6   deleting a specific profile.
7
8 URLs:
9 - `/profiles/`: List all profiles or create a new one.
10 - `/profiles/<int:pk>/`: Retrieve, update, or delete a profile by
11   its primary key.
12 ...
13
14 from django.urls import path
15 from profiles import views
16
17 urlpatterns = [
18     path('profiles/', views.ProfileList.as_view()),
19     path('profiles/<int:pk>/', views.ProfileDetail.as_view()),
20 ]
21
```

Settings:



Results:

All clear, no errors found

# Profiles App

## views.py - Passed

```
1 """
2 Profile Views for handling profile-related API endpoints.
3
4 Includes views for retrieving, listing, and updating user profiles. It also
5 handles custom filtering and ordering options for profiles based on
6 various metrics.
7 """
8
9 from django.db.models import Count
10 from django_filters.rest_framework import DjangoFilterBackend
11 from rest_framework import generics, filters
12 from nomadnarrativesapi.permissions import IsOwnerOrReadOnly
13 from .models import Profile
14 from .serializers import ProfileSerializer
15
16
17 class ProfileList(generics.ListAPIView):
18     """
19         Returns a list of all profiles.
20         No Create view (profile creation handled by Django signals).
21         """
22     queryset = Profile.objects.annotate( # pylint: disable=no-member
```

Settings:



Results:

All clear, no errors found

# Utils

## continents.py - Passed

```
1 """
2 This module provides a mapping of countries to their respective continents.
3 It includes a function to retrieve the continent of a given country.
4
5 The `CONTINENT_MAPPING` dictionary contains the continent as the key and
6 a list of countries as the value. The function `get_continent_by_country`
7 returns the continent name for a given country.
8 """
9
10 CONTINENT_MAPPING = {
11     "Africa": [
12         "Algeria", "Angola", "Benin", "Botswana", "Burkina Faso",
13         "Burundi", "Cabo Verde", "Cameroon", "Central African Republic",
14         "Chad", "Comoros", "Democratic Republic of the Congo", "Djibouti",
15         "Egypt", "Equatorial Guinea", "Eritrea", "Eswatini", "Ethiopia",
16         "Gabon", "Gambia", "Ghana", "Guinea", "Guinea-Bissau",
17         "Ivory Coast", "Kenya", "Lesotho", "Liberia", "Libya",
18         "Madagascar", "Malawi", "Mali", "Mauritania", "Mauritius",
```

Settings:



Results:

All clear, no errors found