

# Cinemat Django project

## PEP8 report

This is a slideshow that showcases the compliance of all python files with PEP8 guidelines

# settings.py

w: 1536 px h: 695 px



## CI Python Linter

```
90     'NAME': (
91         'django.contrib.auth.password_validation.'
92         'UserAttributeSimilarityValidator'
93     ),
94 },
95 {
96     'NAME': (
97         'django.contrib.auth.password_validation.'
98         'MinimumLengthValidator'
99     ),
100 },
101 {
102     'NAME': (
103         'django.contrib.auth.password_validation.'
104         'CommonPasswordValidator'
105     ),
106 },
107 {
108     'NAME': (
109         'django.contrib.auth.password_validation.'
110         'NumericPasswordValidator'
111     ),
112 },
113 ]
114
115 AUTHENTICATION_BACKENDS = [
116     'django.contrib.auth.backends.ModelBackend',
117 ]
118
119 LANGUAGE_CODE = 'en-us'
120 TIME_ZONE = 'UTC'
121 USE_I18N = True
122 USE_TZ = True
123
124 STATIC_URL = '/static/'
125 STATICFILES_DIRS = [os.path.join(BASE_DIR, "static")]
126 STATIC_ROOT = os.path.join(BASE_DIR, "staticfiles")
127 STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'
128
129
130 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
131
```

### Settings:



### Results:

All clear, no errors found

# Actors app

# Actors app / admin.py

w: 1536 px h: 695 px



## CI Python Linter

```
1 from django.contrib import admin
2 from .models import Actor
3
4
5 @admin.register(Actor)
6 class ActorAdmin(admin.ModelAdmin):
7     """
8     Admin configuration for the Actor model.
9
10    Provides a customized interface for managing actor information in the
11    Django admin panel.
12    """
13    list_display = ('name', 'tmdb_id', 'date_of_birth')
14    search_fields = ('name',)
15    list_filter = ('date_of_birth',)
16
```

### Settings:



### Results:

All clear, no errors found

# Actors app / models.py

w: 1536 px h: 695 px



## CI Python Linter

```
1 from django.db import models
2
3
4 class Actor(models.Model):
5     tmdb_id = models.IntegerField(unique=True) # Link to TMDB ID
6     name = models.CharField(max_length=255)
7     date_of_birth = models.DateField(null=True, blank=True)
8     biography = models.TextField(blank=True)
9     # TMDB image path
10    profile_path = models.CharField(max_length=255, blank=True)
11
12    def __str__(self):
13        return self.name
14
```

### Settings:



### Results:

All clear, no errors found

# Actors app / urls.py

w: 1536 px h: 695 px



## CI Python Linter

```
1 from django.urls import path
2 from .views import actor_detail
3
4 urlpatterns = [
5     path("<int:actor_id>/", actor_detail, name="actor_detail"),
6 ]
7
```

### Settings:



### Results:

All clear, no errors found

# Actors app / views.py

w: 1536 px h: 695 px



## CI Python Linter

```
1 import requests
2 from django.shortcuts import render
3 from django.conf import settings
4
5 TMDB_API_KEY = settings.TMDB_API_KEY
6
7
8 def actor_detail(request, actor_id):
9     """Fetches and displays details for a specific actor."""
10     actor_url = (
11         f"https://api.thenoviedb.org/3/person/{actor_id}"
12         f"?api_key={TMDB_API_KEY}"
13         f"&append_to_response=movie_credits,external_ids"
14     )
15     response = requests.get(actor_url)
16
17     if response.status_code != 200:
18         return render(
19             request, "actors/actor_not_found.html", {"actor_id": actor_id}
20         )
21
22     actor = response.json()
23
24     similar_actors = get_similar_actors_by_genre(
25         actor.get("movie_credits", {})
26     )
27
28     # Extract social media links
29     external_ids = actor.get('external_ids', {})
30     social_links = {
31         "imdb": f"https://www.imdb.com/name/{external_ids.get('imdb_id')}"
32         if external_ids.get('imdb_id') else None,
33         "twitter": f"https://twitter.com/{external_ids.get('twitter_id')}"
34         if external_ids.get('twitter_id') else None,
35         "instagram": (
36             f"https://instagram.com/{external_ids.get('instagram_id')}"
37             if external_ids.get('instagram_id') else None
38         ),
39         "facebook": f"https://facebook.com/{external_ids.get('facebook_id')}"
40         if external_ids.get('facebook_id') else None,
41     }
42
```

Settings:



Results:

All clear, no errors found

# Movies app

Movies app has no entities (models) as movies are api calls and not stored in database



# Movies app / urls.py

w: 1536 px h: 695 px



## CI Python Linter

```
1 from django.urls import path, include
2 from .views import (
3     movie_detail,
4     movie_list,
5     toggle_favorite,
6     movie_detail_api,
7     trending_movies,
8     get_favorite_movies
9 )
10
11 app_name = "movies"
12
13 urlpatterns = [
14     path("", movie_list, name="movie_list"),
15     path("<int:movie_id>/", movie_detail, name="movie_detail"),
16     path("toggle_favorite/", toggle_favorite, name="toggle_favorite"),
17     path(
18         "get_favorite_movies/", get_favorite_movies, name="get_favorite_movies"
19     ),
20     path(
21         "api/details/<int:movie_id>/",
22         movie_detail_api,
23         name="movie_detail_api",
24     ),
25     path("trending/", trending_movies, name="trending"),
26     path("reviews/", include("reviews.urls")),
27 ]
28
```

### Settings:



### Results:

All clear, no errors found

# Movies app / views.py

w: 1536 px h: 695 px



## CI Python Linter

```
1 import requests
2 import logging
3 from django.contrib.auth.decorators import login_required
4 from django.shortcuts import render
5 from django.conf import settings
6 from django.http import JsonResponse, HttpResponseBadRequest
7 from users.models import FavoriteMovie, Profile
8 from reviews.models import Review
9 from reviews.forms import ReviewForm
10
11 # Constants for TMDB API
12 TMDB_API_KEY = settings.TMDB_API_KEY
13 TMDB_BASE_URL = "https://api.themoviedb.org/3"
14
15
16 - def get_movie_genres():
17     """Fetch the list of movie genres from TMDB API."""
18     url = f"{TMDB_BASE_URL}/genre/movie/list"
19     params = {"api_key": TMDB_API_KEY, "language": "en-US"}
20     response = requests.get(url, params=params)
21     return response.json().get("genres", [])
22
23
24 - def trending_movies(request):
25     """Fetches trending movies from TMDB and renders the trending page."""
26     url = (
27         f"https://api.themoviedb.org/3/trending/movie/week?"
28         f"api_key={TMDB_API_KEY}"
29     )
30     response = requests.get(url)
31
32 -     if response.status_code != 200:
33         return render(request, "movies/trending.html", {"movies": []})
34
35     movies = response.json().get("results", [])
36
37     return render(request, "movies/trending.html", {"movies": movies})
38
39
40 @login_required
41 - def movie_list(request):
42     """
```

### Settings:



### Results:

All clear, no errors found

# Reviews app

# Reviews app / admin.py

w: 1536 px h: 695 px



## CI Python Linter

```
1 from django.contrib import admin
2 from .models import Review
3
4
5 @admin.register(Review)
6 class ReviewAdmin(admin.ModelAdmin):
7     """
8     Admin configuration for the Review model.
9
10    Provides a customized interface for managing movie reviews in the Django
11    admin panel.
12    """
13    list_display = ('user', 'movie_id', 'rating', 'created_at')
14    search_fields = ('user__username', 'review_text')
15    list_filter = ('user', 'rating')
16
```

Settings:



Results:

All clear, no errors found

# Reviews app / models.py

w: 1536 px h: 695 px



CI Python Linter

```
1 from django.db import models
2 from django.contrib.auth.models import User
3
4
5 class Review(models.Model):
6     """
7     Represents a movie review submitted by a user.
8
9     Attributes:
10         user (User): The user who wrote the review.
11         movie_id (int): The TMDB movie ID for the reviewed movie.
12         review_text (str): The text content of the review.
13         rating (Decimal): The user's rating for the movie (0.0 to 5.0).
14         created_at (datetime): The date and time when the review was created.
15     """
16
17     user = models.ForeignKey(User, on_delete=models.CASCADE)
18     movie_id = models.IntegerField() # TMDB movie ID
19     review_text = models.TextField()
20     # Store user's rating locally, Scale: 0.0-5.0
21     rating = models.DecimalField(max_digits=3, decimal_places=1, default=0.0)
22     created_at = models.DateTimeField(auto_now_add=True)
23
24     def __str__(self):
25         """
26         Returns a string representation of the review.
27
28         Returns:
29             str: A string indicating the reviewer and the movie ID.
30         """
31         return f"Review by {self.user.username} for Movie ID {self.movie_id}"
32
```


Settings:



Results:

All clear, no errors found

# Reviews app / urls.py

w: 1536 px h: 695 px 



## CI Python Linter

```
1 from django.urls import path
2 from .views import submit_review, update_review, delete_review
3
4 app_name = "reviews"
5
6 urlpatterns = [
7     path("submit/<int:movie_id>/", submit_review, name="submit_review"),
8     path("update/<int:review_id>/", update_review, name="update_review"),
9     path("delete/<int:review_id>/", delete_review, name="delete_review"),
10 ]
11
```

### Settings:



### Results:

All clear, no errors found

# Reviews app / views.py

w: 1536 px h: 695 px



## CI Python Linter

```
58         print("Rating successfully submitted to IMDB!")
59     else:
60         print(f"Error submitting rating: {response.json()}")
61
62     return redirect("movies:movie_detail", movie_id=movie_id)
63
64 else:
65     # Prefill form with existing review
66     form = ReviewForm(instance=existing_review)
67
68     movie = get_movie_details(movie_id)
69     return render(
70         request,
71         "movies/movie_detail.html", {"form": form, "movie": movie}
72     )
73
74 @login_required
75 def update_review(request, review_id):
76     review = get_object_or_404(Review, id=review_id, user=request.user)
77
78     if request.method == "POST":
79         data = json.loads(request.body)
80         review.review_text = data.get("review_text", review.review_text)
81         review.rating = data.get("rating", review.rating)
82         review.save()
83
84         return JsonResponse(
85             {"success": True, "review_text": review.review_text})
86     return JsonResponse({"success": False})
87
88
89
90 @login_required
91 def delete_review(request, review_id):
92     review = get_object_or_404(Review, id=review_id, user=request.user)
93
94     if request.method == "POST":
95         review.delete()
96         return JsonResponse({"success": True})
97     return JsonResponse({"success": False})
98
99
```

Settings:



Results:

All clear, no errors found

Users app



# Users app / admin.py

w: 1536 px h: 695 px



## CI Python Linter

```
1 from django.contrib import admin
2 from .models import Profile, FavoriteMovie
3
4
5 @admin.register(Profile)
6 class ProfileAdmin(admin.ModelAdmin):
7     """
8     Admin configuration for the Profile model.
9
10    Provides a customized interface for managing user profiles in the
11    Django admin panel.
12    """
13    list_display = ('user',)
14    search_fields = ('user__username', 'country__name')
15
16
17 @admin.register(FavoriteMovie)
18 class FavoriteMovieAdmin(admin.ModelAdmin):
19     """
20     Admin configuration for the FavoriteMovie model.
21
22    Provides a customized interface for managing user's favorite movies in the
23    Django admin panel.
24    """
25    list_display = ('user', 'title', 'movie_id', 'rating')
26    search_fields = ('user__username', 'title')
27    list_filter = ('user',)
28
```


### Settings:



### Results:

All clear, no errors found

# Users app / models.py

w: 1536 px h: 695 px 



## CI Python Linter

```
1 from django.contrib.auth.models import User
2 from django.db import models
3 from django_countries.fields import CountryField
4
5
6 class Profile(models.Model):
7     """Represents a user profile."""
8
9     user = models.OneToOneField(
10         User, on_delete=models.CASCADE, related_name="profile"
11     )
12
13     def __str__(self):
14         return self.user.username
15
16
17 class FavoriteMovie(models.Model):
18     """Represents a movie favorited by a user."""
19
20     user = models.ForeignKey(User, on_delete=models.CASCADE)
21     movie_id = models.IntegerField()
22     title = models.CharField(max_length=255, default="Unknown Movie")
23     poster_path = models.CharField(max_length=500, blank=True, null=True)
24     release_date = models.CharField(max_length=10, blank=True, null=True)
25     rating = models.FloatField(blank=True, null=True)
26
27     def __str__(self):
28         return f"{self.user.username} - {self.title}"
29
30 class Meta:
31     # Ensures no duplicate movies per user.
32     unique_together = ("user", "movie_id")
33
```

Settings:



Results:

All clear, no errors found

# Users app / urls.py

w: 1536 px h: 695 px



## CI Python Linter

```
1 from django.urls import path
2 from django.contrib.auth.views import LogoutView
3 from .views import (
4     RegisterView,
5     CustomLoginView,
6     profile_view,
7     add_favorite_movie,
8     remove_favorite_movie,
9     change_password,
10    contact_view,
11 )
12
13
14 urlpatterns = [
15     path("register/", RegisterView.as_view(), name="register"),
16     path("login/", CustomLoginView.as_view(), name="login"),
17     path("logout/", LogoutView.as_view(next_page="login"), name="logout"),
18     path("profile/", profile_view, name="profile"), # Logged-in user's profile
19     path("profile/<str:username>/", profile_view, name="user_profile"),
20     path(
21         "add_favorite/<int:movie_id>/",
22         add_favorite_movie,
23         name="add_favorite",
24     ),
25     path(
26         "remove_favorite/<int:movie_id>/",
27         remove_favorite_movie,
28         name="remove_favorite"
29     ),
30     path("change-password/", change_password, name="change_password"),
31     path("contact/", contact_view, name="contact"),
32 ]
33
```

Settings:



Results:

All clear, no errors found

# Users app / views.py

w: 1536 px h: 695 px



## CI Python Linter

```
1 import os
2 import requests
3 from django import forms
4 from django.conf import settings
5 from django.shortcuts import render, redirect, get_object_or_404
6 from django.http import JsonResponse
7 from django.contrib import messages
8 from django.contrib.auth import authenticate, login, update_session_auth_hash
9 from django.contrib.auth.views import LoginView
10 from django.contrib.auth.decorators import login_required
11 from django.contrib.auth.models import User
12 from django.contrib.auth.forms import PasswordChangeForm
13 from django.core.mail import send_mail
14 from django.views.generic import CreateView
15 from django.urls import reverse_lazy
16 from .forms import RegisterForm, ContactForm
17 from .models import FavoriteMovie, Profile
18
19
20 # Home view.
21 def home(request):
22     """Renders the home page."""
23     return render(request, "users/home.html")
24
25
26 # Registration view
27 class RegisterView(CreateView):
28     """
29     Handles user registration using Django's built-in CreateView.
30     Automatically creates a user profile and logs in the
31     user after registration.
32     """
33
34     template_name = "users/register.html"
35     form_class = RegisterForm
36     success_url = reverse_lazy("login")
37
38     def form_valid(self, form):
39         """Saves the user and creates the profile if the form is valid."""
40         user = form.save() # Use the overridden save method from the form
41
42         Profile.objects.get_or_create(user=user)
```

### Settings:



### Results:

All clear, no errors found