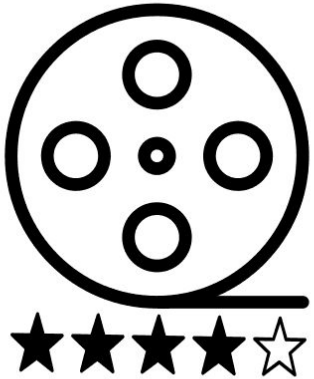


Cinemate Django project PEP8 report

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

settings.py

code institute

CI Python Linter

```
        'NAME': (
            'django.contrib.auth.password_validation.'
            'UserAttributeSimilarityValidator'
        ),
    },
    {
        'NAME': (
            'django.contrib.auth.password_validation.'
            'MinimumLengthValidator'
        ),
    },
    {
        'NAME': (
            'django.contrib.auth.password_validation.'
            'CommonPasswordValidator'
        ),
    },
    {
        'NAME': (
            'django.contrib.auth.password_validation.'
            'NumericPasswordValidator'
        ),
    },
]


AUTHENTICATION_BACKENDS = [
    'django.contrib.auth.backends.ModelBackend',
]

LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True

STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, "static")]
STATIC_ROOT = os.path.join(BASE_DIR, "staticfiles")
STATICFILES_STORAGE = "whitenoise.storage.CompressedManifestStaticFilesStorage"

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Settings:

Results:

All clear, no errors found

Actors app

Actors app / admin.py

code institute

CI Python Linter

```python
from django.contrib import admin
from .models import Actor


@admin.register(Actor)
class ActorAdmin(admin.ModelAdmin):
    """
    Admin configuration for the Actor model.

    Provides a customized interface for managing actor information in the
    Django admin panel.
    """
    list_display = ('name', 'tmdb_id', 'date_of_birth')
    search_fields = ('name',)
    list_filter = ('date_of_birth',)

```

Settings:

Results:

All clear, no errors found

Actors app / models.py

code institute

CI Python Linter

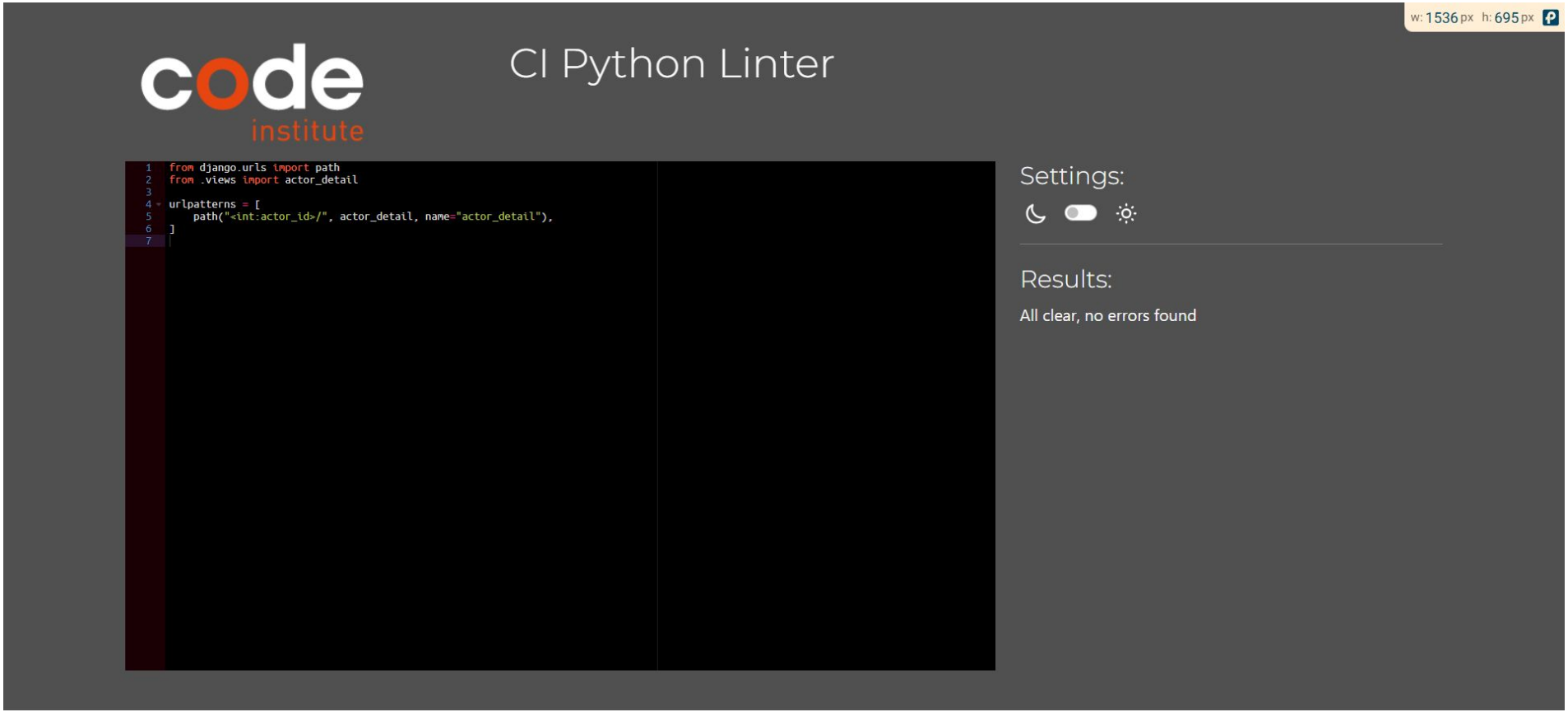
```python
from django.db import models


class Actor(models.Model):
    tmdb_id = models.IntegerField(unique=True)  # Link to TMDB ID
    name = models.CharField(max_length=255)
    date_of_birth = models.DateField(null=True, blank=True)
    biography = models.TextField(blank=True)
    # TMDB image path
    profile_path = models.CharField(max_length=255, blank=True)

    def __str__(self):
        return self.name

```

Settings:

Results:

All clear, no errors found

Actors app / urls.py



Actors app / views.py

code institute

CI Python Linter

```python
import requests
from django.shortcuts import render
from django.conf import settings

TMDB_API_KEY = settings.TMDB_API_KEY


def actor_detail(request, actor_id):
    """Fetches and displays details for a specific actor."""
    actor_url = (
        f"https://api.themoviedb.org/3/person/{actor_id}"
        f"?api_key={TMDB_API_KEY}"
        f"&append_to_response=movie_credits,external_ids"
    )
    response = requests.get(actor_url)

    if response.status_code != 200:
        return render(
            request, "actors/actor_not_found.html", {"actor_id": actor_id}
        )

    actor = response.json()

    similar_actors = get_similar_actors_by_genre(
        actor.get("movie_credits", {})
    )

    # Extract social media links
    external_ids = actor.get('external_ids', {})
    social_links = {
        "imdb": f"https://www.imdb.com/name/{external_ids.get('imdb_id')}"
        if external_ids.get('imdb_id') else None,
        "twitter": f"https://twitter.com/{external_ids.get('twitter_id')}"
        if external_ids.get('twitter_id') else None,
        "instagram": (
            f"https://instagram.com/{external_ids.get('instagram_id')}"
            if external_ids.get('instagram_id') else None
        ),
        "facebook": f"https://facebook.com/{external_ids.get('facebook_id')}"
        if external_ids.get('facebook_id') else None,
    }
```

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

code institute

CI Python Linter

```python
from django.urls import path, include
from .views import (
    movie_detail,
    movie_list,
    toggle_favorite,
    movie_detail_api,
    trending_movies,
    get_favorite_movies
)

app_name = "movies"

urlpatterns = [
    path("", movie_list, name="movie_list"),
    path("<int:movie_id>/", movie_detail, name="movie_detail"),
    path("toggle_favorite/", toggle_favorite, name="toggle_favorite"),
    path(
        "get_favorite_movies/", get_favorite_movies, name="get_favorite_movies"
    ),
    path(
        "api/details/<int:movie_id>/",
        movie_detail_api,
        name="movie_detail_api",
    ),
    path("trending/", trending_movies, name="trending"),
    path("reviews/", include("reviews.urls")),
]

```

Settings:

Results:

All clear, no errors found

Movies app / views.py

w: 1536 px h: 695 px

code institute

CI Python Linter

```python
import requests
import logging
from django.contrib.auth.decorators import login_required
from django.shortcuts import render
from django.conf import settings
from django.http import JsonResponse, HttpResponseBadRequest
from users.models import FavoriteMovie, Profile
from reviews.models import Review
from reviews.forms import ReviewForm

# Constants for TMDB API
TMDB_API_KEY = settings.TMDB_API_KEY
TMDB_BASE_URL = "https://api.themoviedb.org/3"


def get_movie_genres():
    """Fetch the list of movie genres from TMDB API."""
    url = f"{TMDB_BASE_URL}/genre/movie/list"
    params = {"api_key": TMDB_API_KEY, "language": "en-US"}
    response = requests.get(url, params=params)
    return response.json().get("genres", [])


def trending_movies(request):
    """Fetches trending movies from TMDB and renders the trending page."""
    url = (
        f"https://api.themoviedb.org/3/trending/movie/week?"
        f"api_key={TMDB_API_KEY}"
    )
    response = requests.get(url)

    if response.status_code != 200:
        return render(request, "movies/trending.html", {"movies": []})

    movies = response.json().get("results", [])

    return render(request, "movies/trending.html", {"movies": movies})


@login_required
def movie_list(request):
    """
```

Settings:

Results:

All clear, no errors found

Reviews app

Reviews app / admin.py

code institute

CI Python Linter

```python
from django.contrib import admin
from .models import Review


@admin.register(Review)
class ReviewAdmin(admin.ModelAdmin):
    """
    Admin configuration for the Review model.

    Provides a customized interface for managing movie reviews in the Django
    admin panel.
    """
    list_display = ('user', 'movie_id', 'rating', 'created_at')
    search_fields = ('user__username', 'review_text')
    list_filter = ('user', 'rating')
```

Settings:

Results:

All clear, no errors found

Reviews app / models.py

w: 1536 px h: 695 px

code institute

CI Python Linter

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


class Review(models.Model):
    """
    Represents a movie review submitted by a user.

    Attributes:
        user (User): The user who wrote the review.
        movie_id (int): The TMDB movie ID for the reviewed movie.
        review_text (str): The text content of the review.
        rating (Decimal): The user's rating for the movie (0.0 to 5.0).
        created_at (datetime): The date and time when the review was created.
    """

    user = models.ForeignKey(User, on_delete=models.CASCADE)
    movie_id = models.IntegerField()  # TMDB movie ID
    review_text = models.TextField()
    # Store user's rating locally, Scale: 0.0-5.0
    rating = models.DecimalField(max_digits=3, decimal_places=1, default=0.0)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        """
        Returns a string representation of the review.

        Returns:
            str: A string indicating the reviewer and the movie ID.
        """
        return f"Review by {self.user.username} for Movie ID {self.movie_id}"

```

Settings:

Results:

All clear, no errors found

Reviews app / urls.py

code institute

CI Python Linter

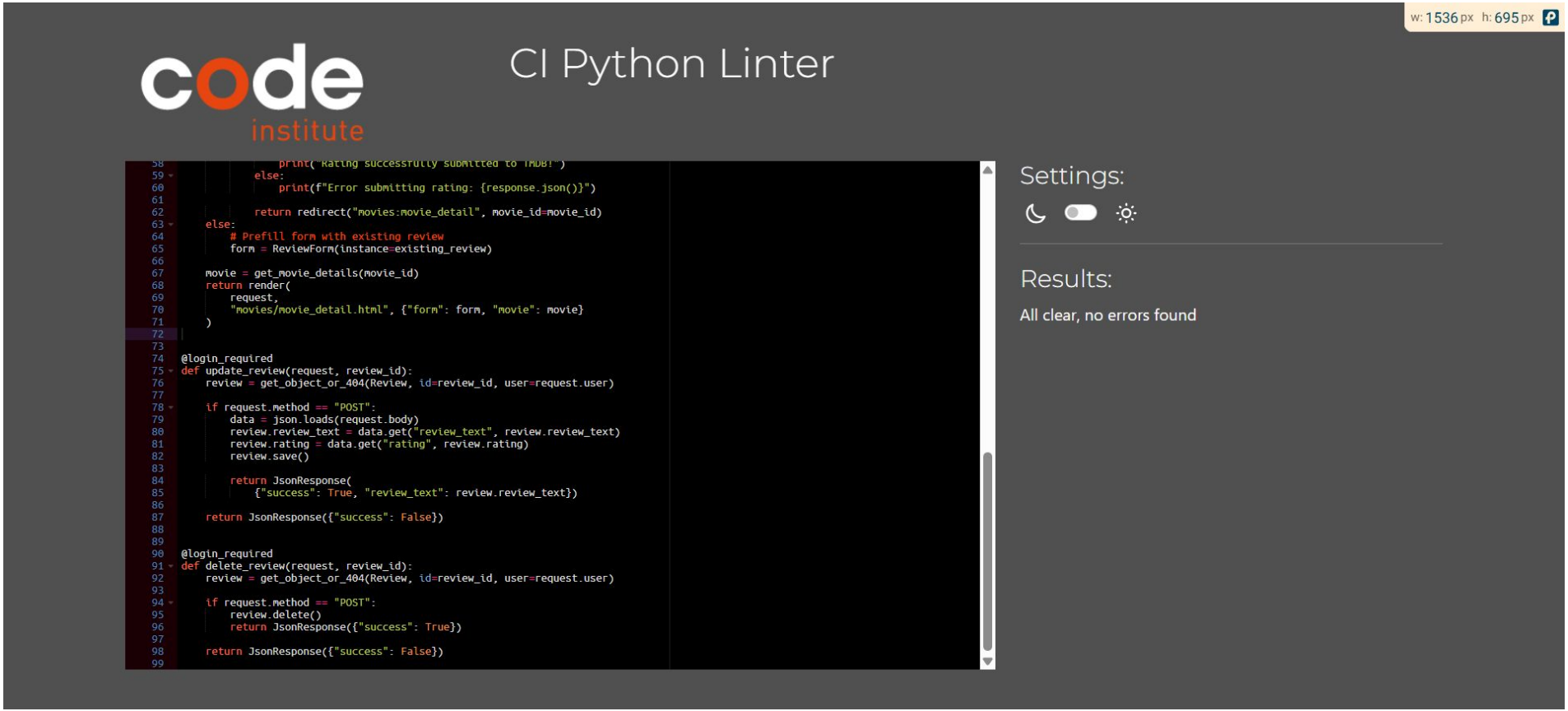
```python
from django.urls import path
from .views import submit_review, update_review, delete_review

app_name = "reviews"

urlpatterns = [
    path("submit/<int:movie_id>/", submit_review, name="submit_review"),
    path("update/<int:review_id>/", update_review, name="update_review"),
    path("delete/<int:review_id>/", delete_review, name="delete_review"),
]

```

Settings:

Results:

All clear, no errors found

Reviews app / views.py



Users app

Users app / admin.py

code institute

CI Python Linter

```python
from django.contrib import admin
from .models import Profile, FavoriteMovie


@admin.register(Profile)
class ProfileAdmin(admin.ModelAdmin):
    """
    Admin configuration for the Profile model.

    Provides a customized interface for managing user profiles in the
    Django admin panel.
    """
    list_display = ('user',)
    search_fields = ('user__username', 'country__name')


@admin.register(FavoriteMovie)
class FavoriteMovieAdmin(admin.ModelAdmin):
    """
    Admin configuration for the FavoriteMovie model.

    Provides a customized interface for managing user's favorite movies in the
    Django admin panel.
    """
    list_display = ('user', 'title', 'movie_id', 'rating')
    search_fields = ('user__username', 'title')
    list_filter = ('user',)

```

Settings:

Results:

All clear, no errors found

Users app / models.py

w: 1536 px h: 695 px

code institute

CI Python Linter

```python
from django.contrib.auth.models import User
from django.db import models
from django_countries.fields import CountryField


class Profile(models.Model):
    """Represents a user profile."""

    user = models.OneToOneField(
        User, on_delete=models.CASCADE, related_name="profile"
    )

    def __str__(self):
        return self.user.username


class FavoriteMovie(models.Model):
    """Represents a movie favorited by a user."""

    user = models.ForeignKey(User, on_delete=models.CASCADE)
    movie_id = models.IntegerField()
    title = models.CharField(max_length=255, default="Unknown Movie")
    poster_path = models.CharField(max_length=500, blank=True, null=True)
    release_date = models.CharField(max_length=10, blank=True, null=True)
    rating = models.FloatField(blank=True, null=True)

    def __str__(self):
        return f"{self.user.username} - {self.title}"

    class Meta:
        # Ensures no duplicate movies per user.
        unique_together = ("user", "movie_id")
```

Settings:

Results:

All clear, no errors found

Users app / urls.py

w: 1536 px h: 695 px

code institute

CI Python Linter

```python
from django.urls import path
from django.contrib.auth.views import LogoutView
from .views import (
    RegisterView,
    CustomLoginView,
    profile_view,
    add_favorite_movie,
    remove_favorite_movie,
    change_password,
    contact_view,
)

urlpatterns = [
    path("register/", RegisterView.as_view(), name="register"),
    path("login/", CustomLoginView.as_view(), name="login"),
    path("logout/", LogoutView.as_view(next_page="login"), name="logout"),
    path("profile/", profile_view, name="profile"),  # Logged-in user's profile
    path("profile/<str:username>/", profile_view, name="user_profile"),
    path(
        "add_favorite/<int:movie_id>/",
        add_favorite_movie,
        name="add_favorite",
    ),
    path(
        "remove_favorite/<int:movie_id>/",
        remove_favorite_movie,
        name="remove_favorite"
    ),
    path("change-password/", change_password, name="change_password"),
    path("contact/", contact_view, name="contact"),
]
```

Settings:

Results:

All clear, no errors found

Users app / views.py

code institute

CI Python Linter

```python
import os
import requests
from django import forms
from django.conf import settings
from django.shortcuts import render, redirect, get_object_or_404
from django.http import JsonResponse
from django.contrib import messages
from django.contrib.auth import authenticate, login, update_session_auth_hash
from django.contrib.auth.views import LoginView
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User
from django.contrib.auth.forms import PasswordChangeForm
from django.core.mail import send_mail
from django.views.generic import CreateView
from django.urls import reverse_lazy
from .forms import RegisterForm, ContactForm
from .models import FavoriteMovie, Profile


# Home view.
def home(request):
    """Renders the home page."""
    return render(request, "users/home.html")


# Registration view
class RegisterView(CreateView):
    """
    Handles user registration using Django's built-in CreateView.
    Automatically creates a user profile and logs in the
    user after registration.
    """

    template_name = "users/register.html"
    form_class = RegisterForm
    success_url = reverse_lazy("login")

    def form_valid(self, form):
        """Saves the user and creates the profile if the form is valid."""
        user = form.save()  # Use the overridden save method from the form

        Profile.objects.get_or_create(user=user)
```

Settings:

Results:

All clear, no errors found