Movie Watchlist API – Django

Step-1:

To install all the required packages. Open Terminal, go to **kickoffs-django-movieswatchlistapi** folder, type "**python3 -m pip install -r requirements.txt**" (or) "**sh install.sh**".

Note: Models are given in models.py file, use serializers.py file to create serializers, use views.py file to create endpoints and use urls.py file inside moviewatchlistapp folder to create all urls.

Return all error message mentioned below with keyword "msg".

Step-2:

Movie Model:

Field Name	Type	Primary Key	Comments
id	Integer	Yes	
title	String		
description	String		
release_year			
genre	String		
is_watched	Boolean		
added_on	DateTime		Default=current datetime

EndPoints:

"/movie":

Post Method:

Get the details (title, description, release_year, genre, is_watched) from the user and store it in the database, then return it as a response with status code 201.

{"title": "Inception", "description": "A skilled thief leads a team into dreams to steal secrets.", "release_year": 2010, "genre": "Science Fiction", "is_watched": True }

Request

{ "id": 1, "title": "Inception", "description": "A skilled thief leads a team into dreams to steal secrets.", "release_year": 2010, "genre": "Science Fiction", "is watched": True, 'added_on': '2025-05-22T11:06:01.639795Z' }

Response

If there is any error while posting the detail, then return the error with status code 400 as a response.

Get Method:

Get genre from the url as a request param and filter all the movie objects which have the given genre from movie model, then return it as a response with status code 200.

Request: localhost:8000/movie?genre= Science Fiction

[{ "id": 1, "title": "Inception", "description": "A skilled thief leads a team into dreams to steal secrets.", "release_year": 2010, "genre": "Science Fiction", "is_watched": True, 'added_on': '2025-05-22T11:06:01.639795Z' }]

Response

If no data available in the movie model then return message "no data available" as a response with status code 400

If no genre is given in the request param then return "genre need to filter" as a response with status code 400.

“movie/<int:id>”:

Patch Method:

Get the detail [is_watched] to be patched from the user and update it in movie object with given id, then return it as a response with status code 200.

{"is watched": False}

Request

{ "id": 1, "title": "Inception", "description": "A skilled thief leads a team into dreams to steal secrets.", "release_year": 2010, "genre": "Science Fiction", "is watched": False, 'added_on': '2025-05-22T11:06:01.639795Z' }

Response

If no data provided in the request body to update or no movie object found for given id, then return "unable to update" as a response with status code 400.

Delete Method:

Get the object with given id from movie model and delete it then return 204 as status code.

If the given id not found, then return message "movie object not found" with status code 400.

Instructions:

Note: Before run and test the application make sure to run the following migration commands.

Migration Commands: Go to kickoffs-django-movieswatchlistapi and type "python3 manage.py makemigrations" and type "python3 manage.py migrate".

To run the application: Open terminal, go to **kickoffs-django-movieswatchlistapi** folder, type as "**python3 manage.py runserver**".

To test the application: Open terminal go to **kickoffs-django-movieswatchlistapi** folder, type as "**python3 manage.py test moviewatchlistapp.tests**".

Check the application for syntax error before you click submit.

To preview the application: Open the browser and type http://localhost:8000/ in address bar and hit enter.

Click **Submit** to complete the test