Date:31.10.25

TASK:12

MOVIE RECOMMENDATION ENGINE

PROBLEM STATEMENT : **CO4, CO5 S3**

In today's world, the number of movies being released is increasing rapidly, making it difficult for users to choose a movie that matches their taste. A Movie Recommendation Engine is required to help users find movies similar to their preferences. The system will take a movie that the user likes as input and suggest other movies with similar characteristics, such as genre, actors, or keywords.The goal is to implement a content-based recommendation system that analyzes the features of movies and recommends those that share the most common attributes with the user's preferred movie. This system will reduce the time a user spends searching for suitable movies and enhance their viewing experience.

AIM :

To develop a Movie Recommendation Engine that suggests movies to users based on the similarity of genres or other features, helping them discover films that match their preferences and enhance their viewing experience.

OBJECTIVE :

1.To create a system that recommends movies based on the similarity of genres, keywords, or features.

2. To allow users to input a movie they like and receive relevant movie suggestions.

3. To implement a content-based filtering algorithm for matching movies.

4. To rank recommended movies based on similarity to the user’s choice.

5. To enhance user experience by reducing the time and effort needed to find interesting movies.

DESCRIPTION :

The Movie Recommendation Engine is a system designed to help users find movies that match their interests. It works by analyzing the features of movies, such as genre, keywords, and other attributes, and then recommending films that are similar to a movie chosen by the user.Using content-based filtering, the system calculates similarity between movies and ranks the recommendations accordingly. This provides users with personalized suggestions, saves time in searching for movies, and enhances their overall viewing experience. The engine can also be scaled to handle larger datasets using advanced techniques like machine learning and vector-based similarity measures.

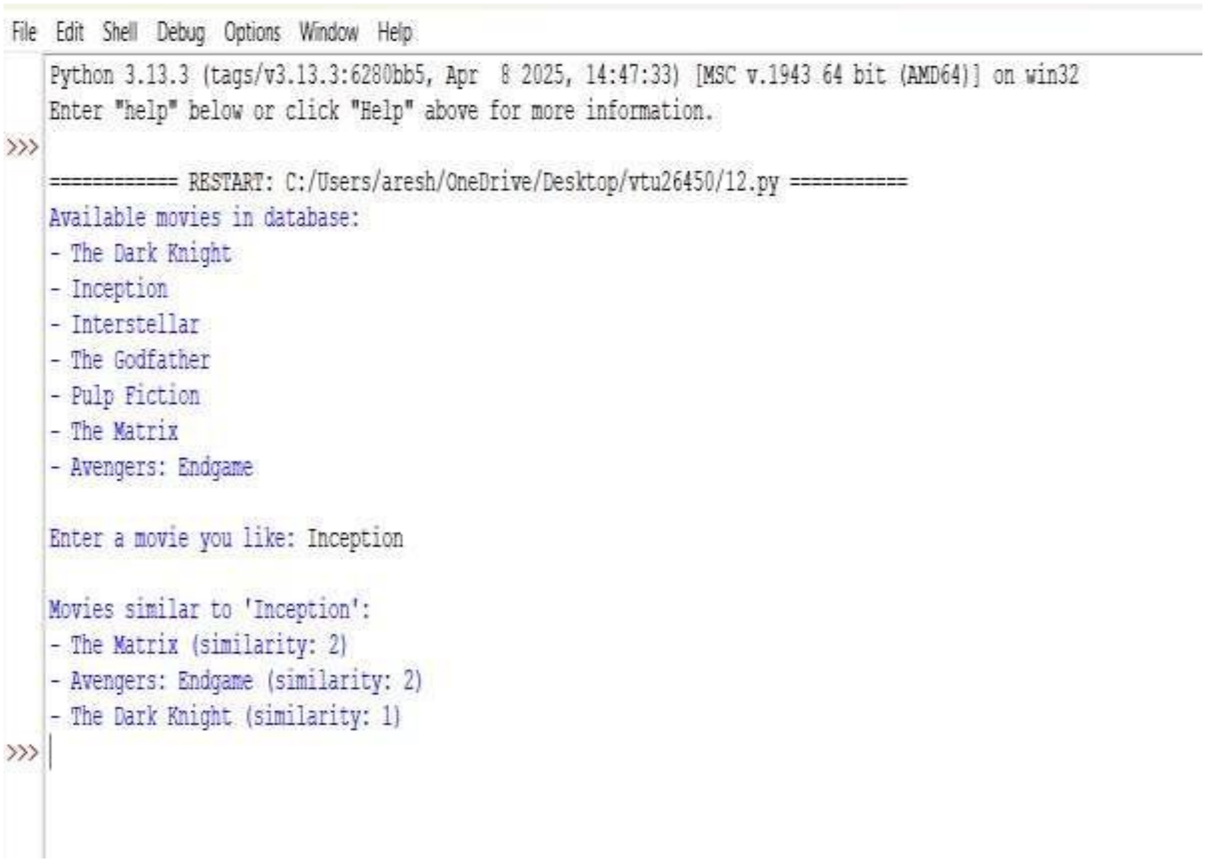
ALGORITHM :

```
Step 1: Start the program.
Step 2: Create a dataset of movies with attributes such as title, genre, keywords, etc.
Step 3: Ask the user to input a movie they like.
Step 4: Search for the input movie in the dataset.
Step 5: Extract the features (e.g., genre, keywords) of the input movie.
Step 6: For each movie in the dataset (except the input movie):
    a) Compare its features with the input movie's features.
    b) Calculate a similarity score (e.g., number of common genres or cosine similarity).
Step 7: Sort all movies based on the similarity score in descending order.
Step 8: Select the top N movies as recommendations.
Step 9: Display the recommended movies to the user.
Step 10: End the program.
```

PROGRAM :

```python
movies = [
    {"title": "The Dark Knight", "genre": "Action, Crime, Drama"},
    {"title": "Inception", "genre": "Action, Sci-Fi, Thriller"},
    {"title": "Interstellar", "genre": "Adventure, Drama, Sci-Fi"},
    {"title": "The Godfather", "genre": "Crime, Drama"},
    {"title": "Pulp Fiction", "genre": "Crime, Drama, Thriller"},
    {"title": "The Matrix", "genre": "Action, Sci-Fi"},
    {"title": "Avengers: Endgame", "genre": "Action, Adventure, Sci-Fi"},]
print("Available movies in database:")
for movie in movies:
    print("-", movie['title'])
def recommend_movie(user_movie_title):
    user_movie_title = user_movie_title.strip().lower()
    user_movie = None
    for movie in movies:
        if user_movie_title in movie['title'].lower():
            user_movie = movie
            break
if not user_movie:
        print("Movie not found in database. Please enter a correct title from the
list.")
        return
   user_genres = set(user_movie['genre'].split(", "))
    recommendations = []
   for movie in movies:
        if movie['title'] != user_movie['title']:
            genres = set(movie['genre'].split(", "))
            similarity = len(user_genres.intersection(genres))
            if similarity > 0:
                recommendations.append((movie['title'], similarity))
           recommendations.sort(key=lambda x: x[1], reverse=True)
      print(f"\nMovies similar to '{user_movie['title']}':")
    if recommendations:
        for movie, sim in recommendations[:3]:
            print(f"- {movie} (similarity: {sim})")
    else:
        print("No similar movies found.")
user_input = input("\nEnter a movie you like: ")
recommend_movie(user_input)
```

OUTPUT :



```
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr  8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
============ RESTART: C:/Users/aresh/OneDrive/Desktop/vtu26450/12.py ===========
Available movies in database:
- The Dark Knight
- Inception
- Interstellar
- The Godfather
- Pulp Fiction
- The Matrix
- Avengers: Endgame

Enter a movie you like: Inception

Movies similar to 'Inception':
- The Matrix (similarity: 2)
- Avengers: Endgame (similarity: 2)
- The Dark Knight (similarity: 1)
>>>
```

CONCLUSION :

The Movie Recommendation Engine suggests movies based on user preferences by analyzing genres and features. It helps users discover relevant movies quickly and enhances their overall viewing experience.