TASK-4

from collections import defaultdict

class MovieRecommendationAI:

def \_\_init\_\_(self):

self.movies = {

1: {"title": "The Shawshank Redemption", "genres": ["Drama"], "rating": 9.3},

2: {"title": "The Godfather", "genres": ["Crime", "Drama"], "rating": 9.2},

3: {"title": "The Dark Knight", "genres": ["Action", "Crime", "Drama"], "rating": 9.0},

4: {"title": "Pulp Fiction", "genres": ["Crime", "Drama"], "rating": 8.9},

5: {"title": "Forrest Gump", "genres": ["Drama", "Romance"], "rating": 8.8},

6: {"title": "Inception", "genres": ["Action", "Adventure", "Sci-Fi"], "rating": 8.8},

7: {"title": "The Matrix", "genres": ["Action", "Sci-Fi"], "rating": 8.7},

8: {"title": "Goodfellas", "genres": ["Biography", "Crime", "Drama"], "rating": 8.7},

9: {"title": "The Silence of the Lambs", "genres": ["Crime", "Drama", "Thriller"], "rating": 8.6},

10: {"title": "Jurassic Park", "genres": ["Adventure", "Sci-Fi", "Thriller"], "rating": 8.1},

11: {"title": "The Lion King", "genres": ["Animation", "Adventure", "Drama"], "rating": 8.5},

12: {"title": "Gladiator", "genres": ["Action", "Adventure", "Drama"], "rating": 8.5},

13: {"title": "The Departed", "genres": ["Crime", "Drama", "Thriller"], "rating": 8.5},

14: {"title": "The Usual Suspects", "genres": ["Crime", "Mystery", "Thriller"], "rating": 8.5},

}

self.genre\_index = self.\_build\_genre\_index()

def \_build\_genre\_index(self):

genre\_index = defaultdict(list)

for movie\_id, movie in self.movies.items():

for genre in movie["genres"]:

genre\_index[genre].append(movie\_id)

return genre\_index

def recommend\_movies(self, genre, n=5):

if genre not in self.genre\_index:

return []

genre\_movies = self.genre\_index[genre]

sorted\_movies = sorted(genre\_movies, key=lambda x: self.movies[x]["rating"], reverse=True)

top\_movies = sorted\_movies[:n]

return [self.movies[movie\_id] for movie\_id in top\_movies]

def interact(self):

print("Welcome to the Movie Recommendation AI!")

print("This AI uses content-based filtering to recommend movies based on genres.")

while True:

print("\nAvailable genres:")

available\_genres = sorted({genre for movie in self.movies.values() for genre in movie["genres"]})

print(", ".join(available\_genres))

genre = input("\nEnter a genre to get movie recommendations (or 'quit' to exit): ").capitalize()

if genre.lower() == 'quit':

print("Thank you for using the Movie Recommendation AI. Goodbye!")

break

if genre not in self.genre\_index:

print(f"Sorry, '{genre}' is not a recognized genre. Please try again.")

continue

recommendations = self.recommend\_movies(genre)

if recommendations:

print(f"\nTop recommended {genre} movies:")

for i, movie in enumerate(recommendations, 1):

print(f"{i}. {movie['title']} (Rating: {movie['rating']}) - Genres: {', '.join(movie['genres'])}")

else:

print(f"Sorry, no movies found in the '{genre}' genre.")

if \_\_name\_\_ == "\_\_main\_\_":

ai = MovieRecommendationAI()

ai.interact()