Usercase1: manual testing + unit test

The cases that I tested manually and they passed

- if the user does not enter a movie title when requested, they will receive a message informing them of the issue and asking for a valid movie title

-if the user enters an incorrect or invalid movie title, they will be notified and asked to provide a correct title

- recommendations are provided for a specific movie; that movie will not be included in the list of recommendations

-the recommended movies will be sorted in descending order of relevance, based on how closely they match the genre and theme of the original movie

The logic used for implementing Use case1

-I created 2 functions: extract_keywords and extract_genres. The first functions parses the keywords column and extracts the list of keyword names. The second function extracts the name of genres for each movie

-Then, I created a movies dataframe that contains three columns: title, genres and keywords

-I used MultiLabelBinarizer to create 2 numerical vectors (vectors_genres and keywords_vector)

- I used a cosine similarity matrix to measure how similar each movie is to every other movie in the dataset

- The recommend_movie function finds movies similar to a given movie title

Usecase2:- manual testing +unit test

-when the user is asked to provide a user id but enters something other than a number, a message will appear notifying them of the error and prompting them to enter a valid numeric user ID

-the movies are displayed in descending order, meaning the most relevant ones appear at the top of the list

The logic used for implementing Use case 2

-trains a SVD model to learn user-movie interactions

-test the model (RMSE, MAE)

-extract latent feature vectors representing user preferences

-I used cosine similarity to find users with similar preferences to the target user

-recommends movies based on similar users

- I initially attempted to implement a KNN algorithm instead of SVD, but I ran into memory allocation issues due to the large size of the ratings database. To mitigate this, I used the line ratings = pd.read_parquet('ratings.parquet') for memory optimization. The KNN implementation is currently commented out in the main function.

-I did not include the database files in the Git folder to avoid using too much memory