**Documentation for the MovieLens**

**Objective**

The goal of this script is to create a user-friendly **movie recommendation system** using Streamlit. The application helps users discover movies by applying filters or entering a specific query. It uses text analysis techniques like vectorization and similarity scoring to recommend the best movies.

**Flow of the Code**

**1. Load Data**

* The script begins by loading three CSV files into pandas DataFrames:
  + **vectorized\_df.csv**: Contains vectorized text features of movies for similarity calculations.
  + **bollywood\_full.csv**: A dataset with complete movie metadata (title, actors, genres, IMDb ratings, etc.).
  + **movies\_refined.csv**: A curated subset of movies (not directly used here).

**2. Set Up the User Interface (UI)**

* **Title**: A welcoming title is displayed to set the tone.
* **Options**: Users choose between two methods for recommendations:
  1. Apply **filters** (genres, ratings, reviews).
  2. Enter a **text query**.

**3. Filters**

* **Predefined Options**:
  + **Genres**: Action, Comedy, Drama, Thriller, Sci-Fi.
  + **Ratings**: Poor, Average, Hit, Blockbuster.
  + **Reviews**: Star ratings from 1 to 5.
* Users select filters from dropdowns or multiselect options.
* Filters narrow down the dataset to movies matching the selected criteria.
* If no valid filters are applied, the user is prompted to choose at least one.

**4. Query Input**

* If the user provides a text query, it’s treated as a search term.
* Example: *"A suspenseful space adventure with thrilling action"*.
* If no query is entered, the selected filters (if any) are used to form a query.

**5. Recommendations**

* **Processing the Query**:
  + The query is vectorized using **TfidfVectorizer** (a text processing tool).
  + Cosine similarity is calculated between the query and the movie descriptions.
  + This identifies the movies most similar to the user's input.
* **Number of Recommendations**:
  + Users select the number of recommendations (1 to 10) using a slider.
* **No Match Scenario**:
  + If no movies match the query, the system displays a message: *"No movies found matching your query."*

**6. Display Results**

* Recommendations are displayed as clickable buttons.
* Clicking on a movie title reveals its details:
  + Title: *Inception*
  + Actors: *Leonardo DiCaprio, Joseph Gordon-Levitt*
  + Genres: *Action, Sci-Fi, Thriller*
  + IMDb Rating: *8.8*
  + Summary: *A skilled thief uses dream invasion technology...*
  + Release Date: *2010-07-16*
  + IMDb Link: [Inception on IMDb](https://www.imdb.com/title/tt1375666/)

**Example of How the Code Works**

**Input**

1. Filters:
   * Genres: *Action, Sci-Fi*.
   * Ratings: *Blockbuster*.
   * Reviews: *5 stars*.
2. Query: *"Time travel and futuristic themes."*

**Flow**

1. The dataset is filtered based on the selected genres, ratings, and reviews.
2. The query is vectorized and compared with the movie descriptions.
3. Movies with the highest similarity scores are identified and recommended.

**Output**

* Top 3 Recommendations:
  1. **Title**: *Interstellar*
     + Actors: *Matthew McConaughey, Anne Hathaway*.
     + Genres: *Action, Sci-Fi, Drama*.
     + IMDb Rating: *8.6*.
     + Summary: *Explorers travel through a wormhole to save humanity.*
     + Release Date: *2014-11-07*.
  2. **Title**: *Edge of Tomorrow*
     + Actors: *Tom Cruise, Emily Blunt*.
     + Genres: *Action, Sci-Fi*.
     + IMDb Rating: *7.9*.
  3. **Title**: *Inception*.

**Objectives Achieved**

1. **Personalized Movie Recommendations**: Based on user filters or custom queries.
2. **Easy-to-Use Interface**: Simple dropdowns and input fields for user interaction.
3. **Rich Metadata**: Provides movie details, including IMDb ratings and clickable links.
4. **Interactive Experience**: Users can explore movies effortlessly.

This application is designed to make movie discovery engaging and enjoyable! 🎥