REVIEW AGGREGATOR

GROUP MEMBERS:

1. 200050003 - Aditya Jain
2. 200050088 - Nikhil Manjrekar
3. 200050127 - Sankalp Parashar
4. 200050156 - Vidit Goel

OVERVIEW:

Aim of the project is to design a review aggregator app for movies and TV series that provides users with a comprehensive overview of reviews and ratings for movies from multiple sources.

The app will have features that allow users to search for movies, view their ratings and reviews from various sources, and sort them based on their preferences. The users can also maintain a watchlist of movies to watch next.

Additionally, the app will provide users with recommendations based on their viewing history, preferences and also mood. Furthermore, the app should have a user-friendly interface and be accessible across multiple platforms and devices.

IMPLEMENTATION DETAILS:

We intend to use React for Frontend, Nodejs for Backend with Postgresql database.

To do this, we need to maintain a database consisting of movie information. This database would be updated in regular intervals. Updates to the database can be done through manual addition of movies or through calls to the API of movie rating sites like IMdB. There's quite a bit to be decided here because many movie API services are paid, but we have found free services that we can use for this purpose.

A prototype of the database schema is given below:

Movie(m_id, m_name, genre, tags)
Links(m_id, movie_id for other sources)
Review(source, m_id, r_id, review (string))
Rating(source, m_id, r_id, rating (int))
Person(ID, Name)
Works(person_id, work, m_name)
Trending(movie_id, rating_count)

Here m_id and m_name are movie ID and movie Name. Most of the tables are self explanatory. The table Person contains names and ID of people associated with movies (Actors/Directors). Table Works contains the work of the person (actor or director) in movies. This is required when we want to show movies involving a certain actor or director. We can also have a table for currently trending movies which can be updated on regular intervals. The rating_count attribute is the number of ratings given to the movie in say last 2 weeks. The movies for which this value is high would be the trending movies. Note that this schema is not final. We will include more attributes and tables as we develop features.

For the watchlist feature, we will have the watchlist attribute in the User table, containing the list of movie IDs in the watchlist.

The Frontend will support pages for homepage, individual movies, and individual directors or actors. There would also be leaderboards for top N movies of all time or top N movies of a given genre, on the basis of popularity or rating.

To obtain a set of recommendations for a user, simple algorithms can be used, like finding patterns such as common genre or actor/director in the last K movies watched by the user, and suggesting popular or trending movies following that pattern.

For login purposes we would follow the outline specified by one of the previous database assignment, including sessions management and bcrypt.