MovieNet: A Social Network for Movie Enthusiasts

445 Course Project

MovieNet is a social network for movie enthusiasts, containing a database of movies, actors, directors, etc., and a social network of movie enthusiasts. Users of MovieNet can search for movies, actors, and directors. They can also rate movies and write reviews of movies. They can further be friends with each other based on movie interests or other similarities. An online movie store can pay to join MovieNet and publish ads that are customized to each user based on her movie interest. This application bears similarity to both the IMDB database (http://www.imdb.com/) and facebook (http://www.facebook.com), but not limited to the services that these applications provide.

The MovieNet that you design and develop should meet the following requirements: (i) minimum functionalities, (ii) good performance for common user operations, and (iii) a significant extension the project, e.g., an integration of a movie database with a social network or an online movie store. Grading of the project combines all three requirements using the scheme below.

Grading scheme:

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>60%</td>
<td>Correct implementation of minimum functionalities listed in Section 1</td>
</tr>
<tr>
<td>10%</td>
<td>Performance of required features</td>
</tr>
<tr>
<td>30%</td>
<td>A major extension of the project</td>
</tr>
</tbody>
</table>

The three requirements are explained in detail in Sections 1, 2, 3, respectively. The data set that we provide for building the movie database is described in Section 4.

1 Minimum Functionality Requirements

The minimum requirements regard the Internet movie database that you are asked to build. The Internet movie database must support the following features.

1.1 Searches

Your application should support three types of searches. These searches can be performed by any user without having to login (i.e., without being registered on the site).

A. Simple searches based on properties of movies. For this category, your application should support the following queries:
   - Find all movies that have ratings in the input rating range.
   - Find all movies that were released in the input year range.
   - Find movies with title given as input. Users should be able to search for both exact matches and substring matches (using the LIKE construct in SQL).

B. Complex searches involving multiple entities. The following queries involve the correlation of two or more entities.
- Find movies with a particular actor, actress, or director given as input. Users should be able to search for both exact matches and substring matches (using the LIKE construct).
- Find movies that have all the input actors (and/or actresses) in it.
- Find all actors and/or actresses who have worked with a given director.

C. Top K searches
- Show top k (k≥1) movies based on average user rating. In case of a tie, show any top k movies.

1.2 User Login, Rating and Reviews of movies
As mentioned before, users can rate a movie in the database. The database stores the average rating of a movie based on up-to-date rating information as well as all user reviews about a movie.

User login
For users to rate movies or write reviews, they need to register to the site and create logins for themselves. After they login to the site with the correct user name and password, they can rate and review movies.

Rating of a movie
A user can rate a movie in the range of 1-10, and is allowed to rate a movie at most once. Every time a new rating is generated, the average rating for the movie is updated in the database. (Hint: you may want to store the average rating of a movie and the count of users who have rated this movie so far for such updates). It is up to you, the owner of the database, to decide if you allow a user to update/delete his rating of a movie.

Reviews of a movie
Every time a user reviews a movie, the review is stored in the database. Note that users are allowed to review a movie multiple times. Again, it is up to you if you allow users to update/delete their reviews.

1.3 SQL Interface for the Administrator
The database administrator is a special user with the user name “admin”. She has unrestricted access to all the data in the database.

Searches by the Administrator
Besides the search interface for regular users (i.e., using text boxes on a search web page), you website should also support the direct input of SQL queries for use by the administrator.

Updates by the Administrator
Information about the movies, actors, actresses, directors, and mpaa ratings in the movie database can only be updated with an administrator login. (Hint: you can use the same SQL interface for both searches and updates by the administrator.)
1.4 Constraints

While many of the constraints to which you need to pay attention are described above, they are summarized and re-iterated below.

**Loading the database**
- At the beginning, the database is loaded with initial values from the dataset supplied with the project (explained more in Section 4). In the loading process, you may also want to create values of other attributes that are in the schema of your database but not in the initial data sets (e.g., the movie id). If an attribute is present in more than one table in your schema (e.g., the movie id may appear in multiple tables), please make sure that values of this attribute are consistent across all tables.

**Updating the database**
- After the initial loading, only the administrator is allowed to add tuples representing new movies, new actors, new actresses, or new directors to the relevant tables, and to update or delete existing tuples about movies, actors, actresses, and directors.
- The database can also be updated with the information of user login when a user registers to the site, and the information of user ratings and reviews.

**Need of user login**
- A user can search the database without having to login.
- But login is needed to be able to rate movies or write reviews.

**Ratings and reviews of movies**
- To ensure fair rating, each user is allowed to rate a movie only once.
- In addition, *a user under the age of 18 is only allowed to rate and write reviews for certain movies based on the mppa-rating of a movie and the user’s age*. In this project, please use the following table as the guideline for deciding who can rate which movies as well as write reviews for these movies.

<table>
<thead>
<tr>
<th>Rating</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>Anyone can rate the movie</td>
</tr>
<tr>
<td>PG</td>
<td>Children under 10 years of age cannot rate the movie</td>
</tr>
<tr>
<td>PG-13</td>
<td>People under 13 years of age cannot rate the movie</td>
</tr>
<tr>
<td>R</td>
<td>People under 17 years of age cannot rate the movie</td>
</tr>
<tr>
<td>NC-17</td>
<td>People under 18 years of age cannot rate the movie</td>
</tr>
</tbody>
</table>

2 Performance Requirements

Your application is expected to run with reasonable performance. While there are no specific requirements in terms of response time, most operations on the website, including user registration, user login, searches, and updates, should be completed with no obvious delay.
If you observe significant delay in common operations, it is time for you to review your design and implementation to identify the causes of slow performance. As general guidelines, you may want to consider the following things:

- Can you refine the database schema to improve performance? You may want to apply the theory of normalization that we learned in class.
- Can you create additional indexes to improve performance? Refer to the lecture on physical design and tuning for a detailed discussion of this topic. The SQL command for creating an index is `CREATE INDEX index_name ON table_name (column_name(s))`.
- Did you implement the functionalities of your application in the right place of your program? For example, a join between two tables should be performed in the database backend. Performing the join in your PHP code is a mistake—even though you may still provide correct answers, the performance of your application will evitably suffer.

### 3 Project Enhancement

Your application should have a significant extension beyond the basic functionalities described above. The following list includes several ideas for the project enhancement. Students can feel free to refine these ideas and implement different variants of them—this is the part of the project where you have the most flexibility. If you believe that you have a brilliant new idea, please come to discuss with the instructor. If your idea is deemed feasible for implementation within a semester and to involve a similar amount of work as other ideas, it will qualify as an extension to the basic functionalities.

To facilitate the extension of your project, the 445 staff will provide you with a large user table with synthetic user data including the following attributes:

- email address (unique),
- user name,
- password,
- age,
- gender,
- location,
- school,
- movie interest, which is described by a list of genres of interest to the user and a list of her favorite movies.

**Social networking among movie enthusiasts**

Users of the movie database would like to be friends with each other based on their movie interests, educational background, age and location of residence, etc. In addition, friends would like to see each other’s activities, such as the movie seen recently, the rating of the movie, the review of the movie, a movie wish list, the movie planned to see next, and so on. One feature of the MovieNet is to send in-real updates about such friend activities. That is, the updates about a MovieNet user will show up immediately in the windows of her friends online. This update feature is similar to that supported by facebook, youtube, windows live etc., offering a great opportunity for users to know what friends are doing.

In your implementation, you should develop an extension of the database to generate the
structure of the social network (i.e., randomly selecting 5-10 friends for each user), store the
updates about each user, and display the updates on her friends’ side soon after the updates occur.

**Customized advertisements**
Online advertisements are important for an online movie store to make profit, but MovieNet
users would only want to receive advertisements that match their interests. Advertising based on
user movie preference, her movie search history, or her web page visit history has been a popular
strategy. Your application can even use the user friends’ movie interests to help online
advertising.

In your implementation, you might need to store user movie search history and web page visit
history (in addition to the movie preference already stored in her profile), do some analysis of all
of this information in order to figure which movies may interest the user, and display those
movies in the user window. If the advertising strategy is purely based on user search/visit history,
it probably will be random at the beginning and become more relevant as information gets
accumulated over time. If a customized ad indeed interests the user (e.g., she clicked on it), the
online store will pay MovieNet for this ad. When the user actually requests to see the movie,
then the online store makes a profit.

Below are some other (smaller) ideas. Please feel free to integrate them with the extensions
outlined above or to significantly expand these ideas into a major extension of your project.

- You can create a user community where a user may maintain a list of his favorite movies and
  a to-watch list. Other users can view these lists. You can further allow users to find the
  common set of favorite movies among them.
- You may also want to integrate your application with other data sources online. Examples
  include (1) integrating your application with movie reviews from “imdb” or “new york
times”, (2) fetching posters and trailers of movies from their official websites, (3)
  integrating your application with showtime tables online.
- Your application can also have a notification service, e.g., using XML/RSS feeds to report to
  interested users the latest addition to the movie database or top 10 movies this week.
- You may want to augment the security of your application, e.g, developing mechanisms to
  protect the database from a variety of attacks.

4 Data Sets
You will be provided with several data sets in the form of text files. It is an important initial
step of your project to parse the data in those files and load it into the database that you will
create based on your designed schema. After the data is loaded into the database, your movie
database is ready to accept requests for search, update, and deletion.

The format of each data set is explained in order below. All of them are tab-delimited.

1. movies.txt
2. directors.txt
3. actors.txt
4. actresses.txt
5. mpaa_ratings.txt
6. ratings.txt
7. genres.txt
8. users.txt

movies.txt
Each line contains two fields, separated by a tab. These two fields combined uniquely identify a movie. In cases in other files where a movie is referenced, both its name and year are listed.
- First field is the movie name.
- Second field is the movie year.

directors.txt
The information about a director is presented by the director’s name in the first line and each movie directed by him/her in a separate, subsequent line. Any line that does not begin with a tab is a director’s name. All subsequent lines that begin with a tab are movies directed by this director, until you hit another line that does not begin with a tab.

actors.txt
This file is in the exact same format as directors.txt.

actresses.txt
This file is in the exact same format as directors.txt.

mpaa_ratings.txt
Each line contains three fields, separated by tabs.
- First field is the movie name.
- Second field is the movie year.
- Third field is the movie MPAA rating.

ratings.txt
Each line contains four fields, separated by tabs.
- First field is the movie name.
- Second field is the movie year.
- Third field is the count of the number of people who have rated the movie. You will need this to calculate the rating of a movie as more people rate it.
- Fourth field is the actual rating, averaged across all the ratings from all the people who rated the
movie.

**genres.txt**
Each line contains three fields, separated by tabs.
- First field is the movie name.
- Second field is the movie year.
- Third field is the movie genre.

**users.txt**
The user data file has a variable number of lines per record. Each record is terminated with a line containing only the pound (#) symbol. Here is the format:

```
<name> <gender> <age> <email> <password> <state> <school>
<genre1> <genre2> ... <genreN>
<favoritemovie1name> <favoritemovie1year>
<favoritemovie2name> <favoritemovie2year>
...
<favoritemovieNname> <favoritemovieNyear>
#
```

The first line consists of 7 tab-delimited fields:
- First field is the user’s name (first-space-last).
- Second field is the user’s gender.
- Third field is the user’s age.
- Fourth field is the user’s email address. This is a uniquely-identifying attribute, so use it as the user’s login name.
- Fifth field is the user’s plaintext password (yes, they are unrealistic).
- Sixth field is the user’s state.
- Seventh field is the user’s school. Not all users have schools (NULL if no school).

The second line consists of a variable number of tab-delimited entries. Each entry corresponds to a genre that this user likes. Note that this line could be blank, which means this user likes no genres.

Any additional lines in the record (there aren’t necessarily any) are movies, presented in the familiar two-field tab-delimited format:
- First field is the movie name.
- Second field is the movie year.
A sample data set is available on the course web site under the “projects” page. The text files for this project, which are much larger than the sample files, will be uploaded to the “/usr/net/ftp/pub/cs445/f2009/project/” directory on the edlab machines.