URL: http://flip3.engr.oregonstate.edu:8374/movies
Admin Password for Group Updates: Password21

Members:
[redacted]
Kevin Le
[redacted]

Title:
Movie Vault

Project Step 7 - Final Report

A. Feedback and Changes

Originally, our relational schema had two separate tables for critic and member. The only difference between the two was that the critic had to include the company that they worked for. However, by having two separate tables, a critic and a member could have the same email. After much discussion, we decided that we should only have the member table and remove the critic table. The member table will have a company option that is optionally. If a user leaves this field blank, then they are a normal member. If the company field is filled in, then they are a critic. This resolves the issue of a member and a critic having the same email. It also made it easier for searching up users, because there was only one table instead of two.

With the deletion of the Critic table, we needed another table to meet the requirements. We decided to create a Groups table. Members can join a group so they can discuss films with one another. They can also create groups and leave groups if they would like. By adding the Groups table, we met the requirements of the database. In order to update the name of the group or delete a group, the user must enter an admin password that we created. We implemented this because we believed that only admins should have the ability to do these functions.

Another change we made was with the user interface. In one of the peer reviews, they suggested that we add a navigation bar in the CSS. We did just that and in our current version, there is a navigation bar that allows users to quickly go from the home page to their profile page to the search bar, etc.

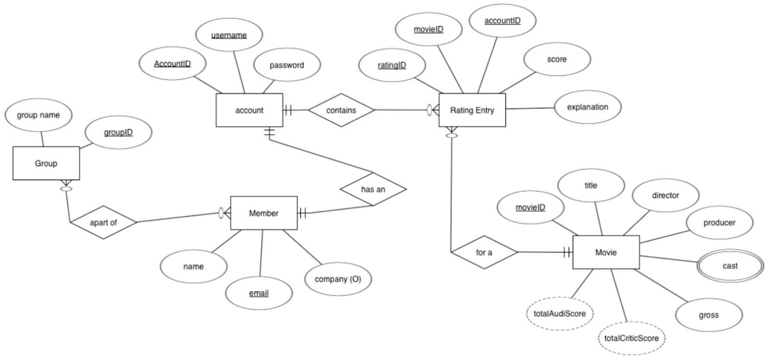
Kevin did the profile page, all the rating entry pages, and the login/signup pages. Michael did half of the group pages and the single movie pages. Sam did the other half of the group pages. Kade did the search functionality and the all movies page.

B. Project Outline

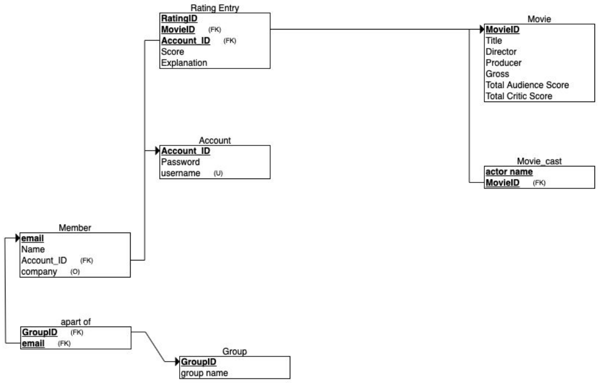
A user can sign in or create an account. A user signs in by entering a username and password. A user can create an account by providing a name, username, password, e-mail, and a company if they have one. If a member has a company listed, then they are a critic, if it is NULL, then they are a normal member. Both are uniquely identified with their email and account ID. A member can be a part of a group, so that they can discuss movies with other members. They can join a group, leave a group, create a group, delete a group, and/or edit the name of the group. The latter 2 require an Admin password which has been hardcoded into the js code. A member and a critic has an account that contains all their rating entries. An account is uniquely identified by the accountID. An account also has a unique username associated with it. A user submits a rating by going to their account and clicking on "Write a Review". An account can only have one rating entry for a specific movie. An account can have multiple rating entries, as long as each one is for a different movie. A movie has many rating entries and has two scores, one for critic ratings and one for member ratings. Our website has a list of movies and when a user clicks on a movie, more details about that movie is provided.

C. Database Outline

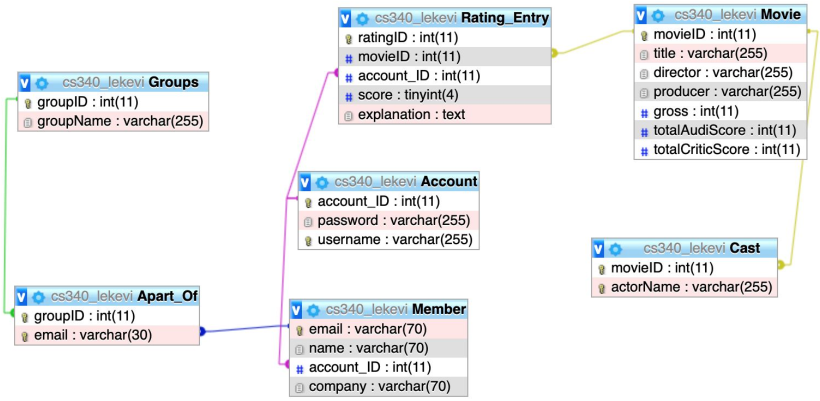
ERD:



Relational Schema:

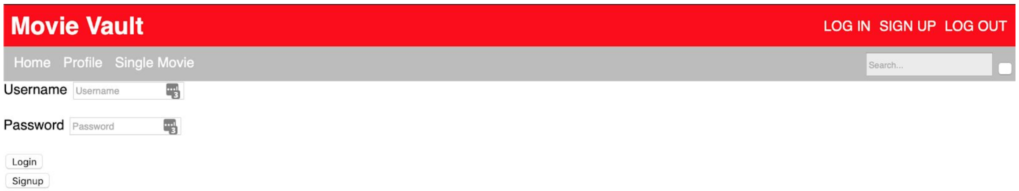


PHP:



D. Screenshots of UI Pages

1.) Read ACCOUNT



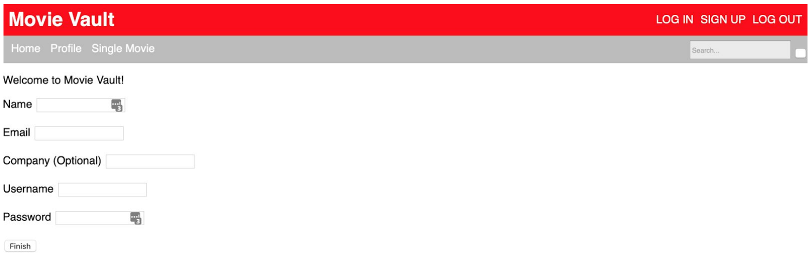
Username Username

Password Password

Queries:

1. SELECT * From Account

2.) Create ACCOUNT/Create MEMBER



Welcome to Movie Vault!

Name

Email

Company (Optional)

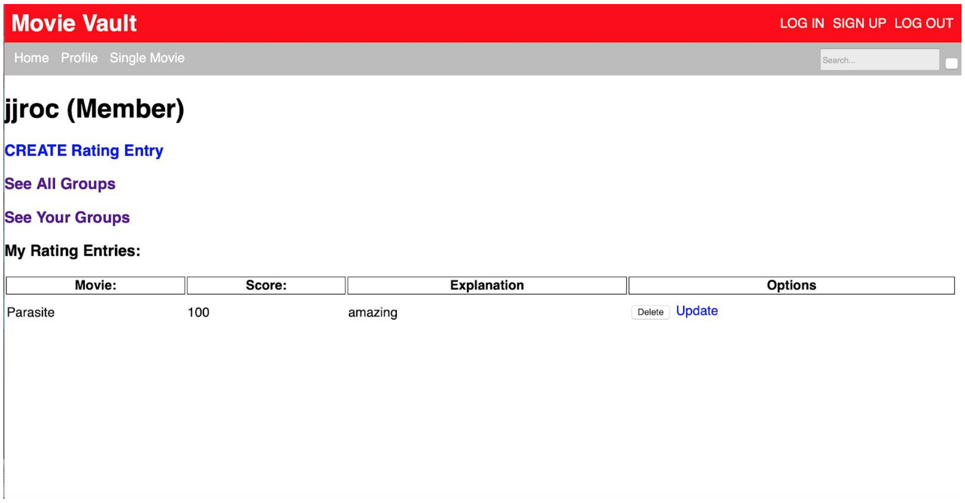
Username

Password

Queries:

1. SELECT * From Account INNER JOIN Member on Account.account_ID = Member.account_ID
2. CALL createAccount (email, name, company, username, password)

3.) Read/Delete RATING_ENTRY



jjroc (Member)

CREATE Rating Entry

See All Groups

See Your Groups

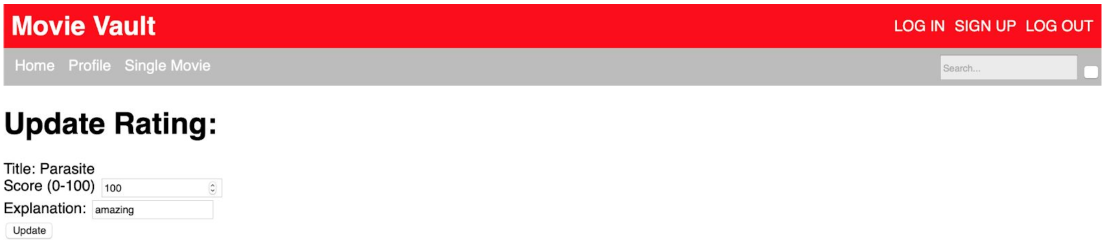
My Rating Entries:

Movie:	Score:	Explanation	Options
Parasite	100	amazing	Delete Update

Queries:

1. SELECT title, score, explanation, ratingID From Rating_Entry Inner Join Movie on Rating_Entry.movieID=Movie.movieID WHERE account_ID = $currentID
2. SELECT username, company From Account a INNER JOIN Member m on a.account_ID=m.account_ID Where a.account_ID = $currentID
3. DELETE FROM Rating_Entry WHERE ratingID = ?

4.) Read/Update RATING_ENTRY



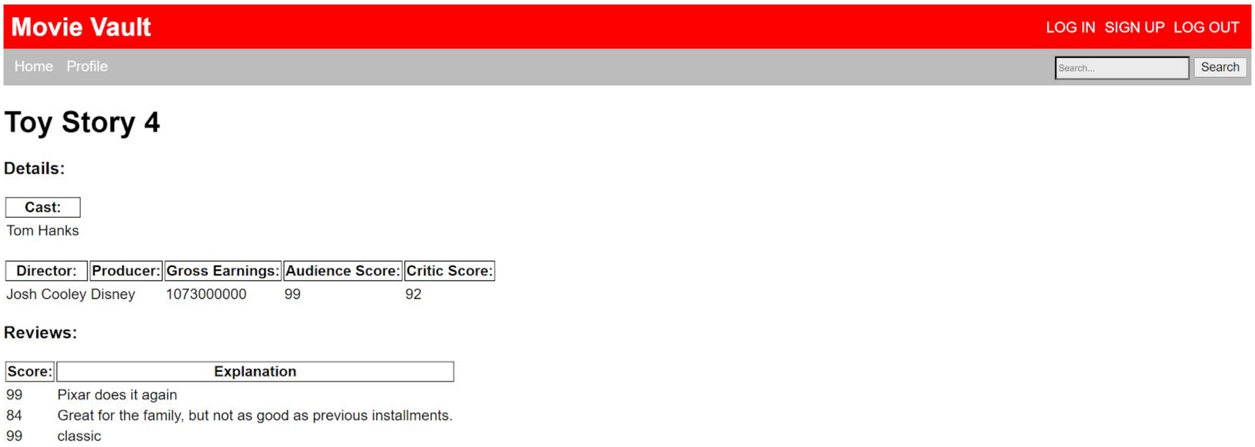
Update Rating:

Title: Parasite
Score (0-100) 100
Explanation: amazing

Queries

1. UPDATE Rating_Entry SET score=?, explanation = ? WHERE ratingID=?
2. SELECT title, score, explanation, ratingID From Rating_Entry Inner Join Movie on Rating_Entry.movieID=Movie.movieID Where ratingID = ?

5.) Display MOVIE/Rating_ENTRY



Toy Story 4

Details:

Cast:
Tom Hanks

Director:	Producer:	Gross Earnings:	Audience Score:	Critic Score:
Josh Cooley	Disney	1073000000	99	92

Reviews:

Score:	Explanation
99	Pixar does it again
84	Great for the family, but not as good as previous installments.
99	classic

Queries:

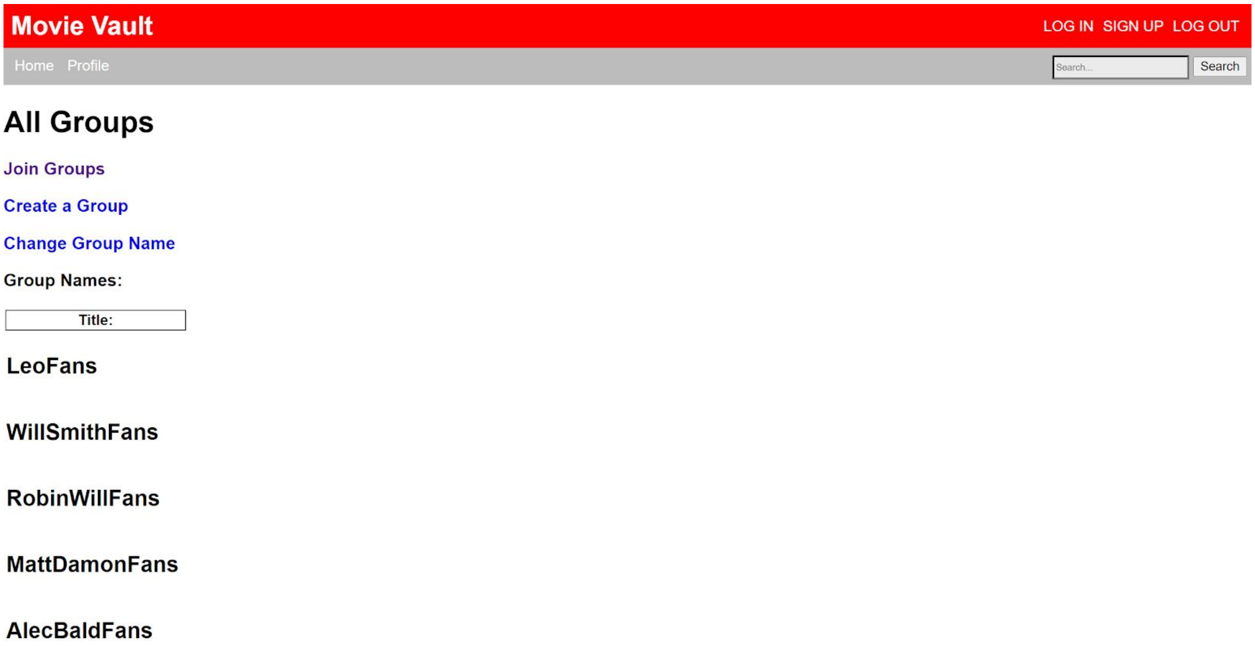
1. "SELECT score, explanation From Rating_Entry Inner Join Movie on Rating_Entry.movieID = ? WHERE Movie.movieID = ?"

2. "SELECT director, producer, gross, totalAudiScore, totalCriticScore From Movie WHERE movieID = ?"

3. "SELECT title From Movie WHERE movieID = ?"

4. "SELECT actorName From Cast WHERE movieID = ?"

6.) Display GROUPS



All Groups

Join Groups

Create a Group

Change Group Name

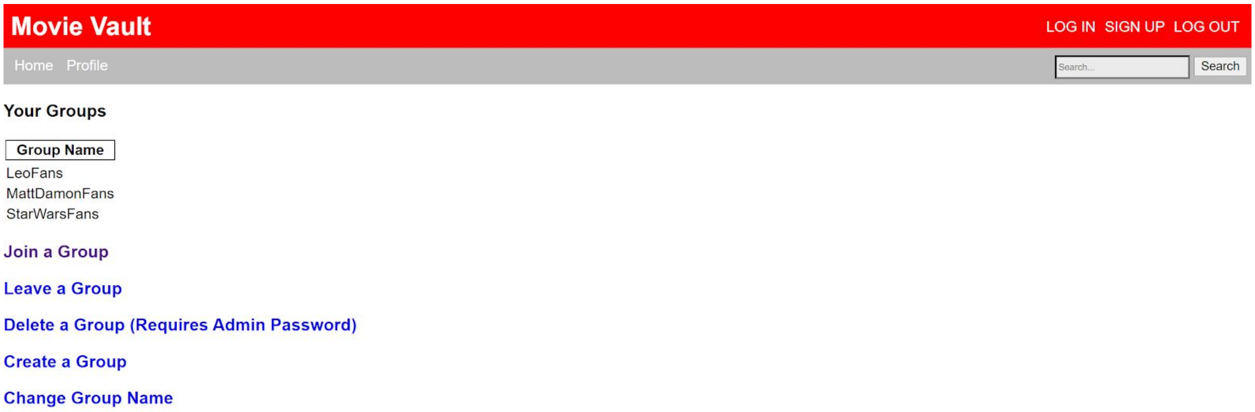
Group Names:

Title:
LeoFans
WillSmithFans
RobinWillFans
MattDamonFans
AlecBaldFans

Queries:

1. "SELECT groupName, groupID From Groups"

7.) Read APART_OF



Your Groups

Group Name
LeoFans
MattDamonFans
StarWarsFans

Join a Group

Leave a Group

Delete a Group (Requires Admin Password)

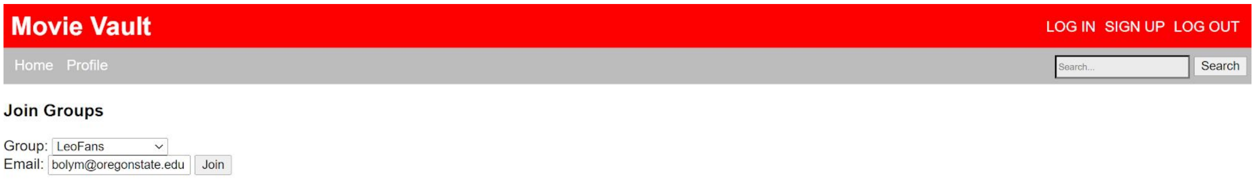
Create a Group

Change Group Name

Queries:

1. "SELECT groupName From Groups AS g INNER JOIN Apart_Of AS a ON g.groupID = a.groupID LEFT JOIN Member AS m ON a.email = m.email WHERE m.account_ID = ?"

8.) Insert APART_OF



Join Groups

Group: LeoFans
Email: bolym@oregonstate.edu

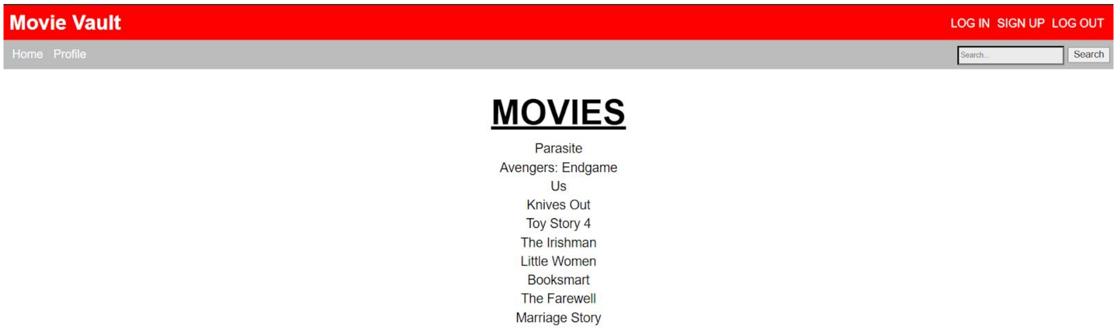
Queries:

1. "SELECT groupID as id, groupName From Groups"

2. "SELECT email FROM Member INNER JOIN Account ON Member.account_ID WHERE Member.account_ID = ?"

3. "INSERT INTO Apart_Of (groupID, email) VALUES (?,?)"

9.) Read MOVIES



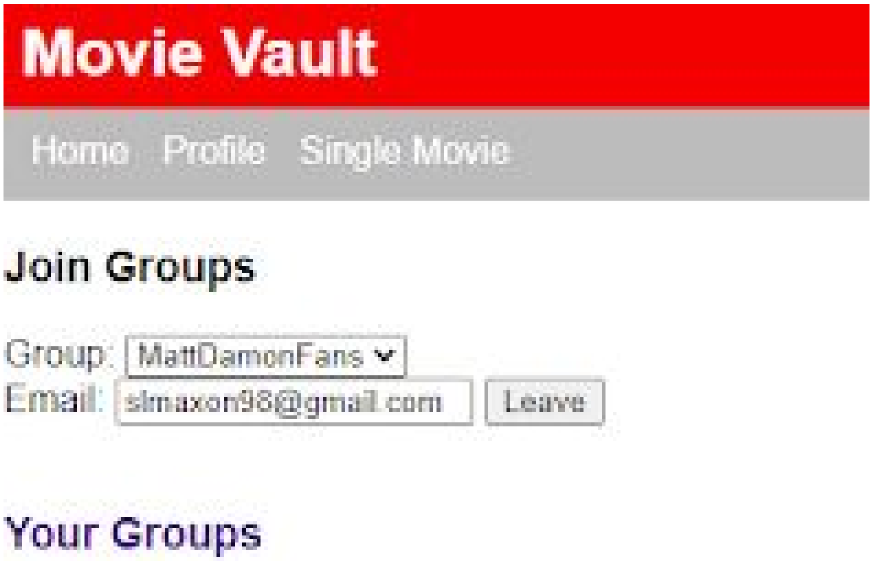
MOVIES

Parasite
Avengers: Endgame
Us
Knives Out
Toy Story 4
The Irishman
Little Women
Booksmart
The Farewell
Marriage Story

Queries:

1. "SELECT title, movieID From Movie"

10.) DELETE/READ APART_OF



Join Groups

Group: MattDamonFans

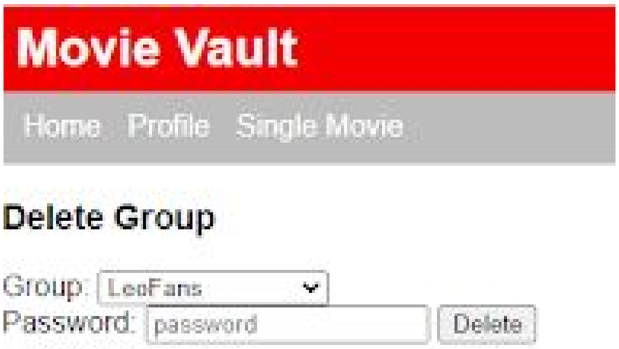
Email: slmaxon98@gmail.com

Your Groups

Queries:

1. "SELECT groupID as id, groupName From Groups NATURAL JOIN Apart_Of NATURAL JOIN Member WHERE account_ID = ?"
2. "SELECT email FROM Member INNER JOIN Account ON Member.account_ID WHERE Member.account_ID = ?"
3. "DELETE FROM Apart_Of WHERE groupID = ? and email = ?"

11.) DELETE/READ GROUPS/APART_OF



Delete Group

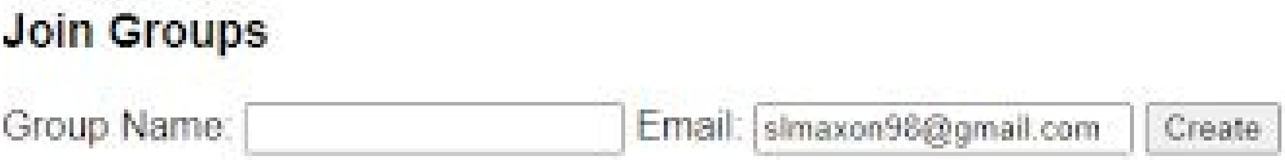
Group: LeoFans

Password: password

Queries:

1. "SELECT groupID as id, groupName From Groups"
2. "DELETE FROM Apart_Of WHERE groupID = ?"
3. "DELETE FROM Groups WHERE groupID = ?"

12.) INSERT APART_OF/GROUPS



Join Groups

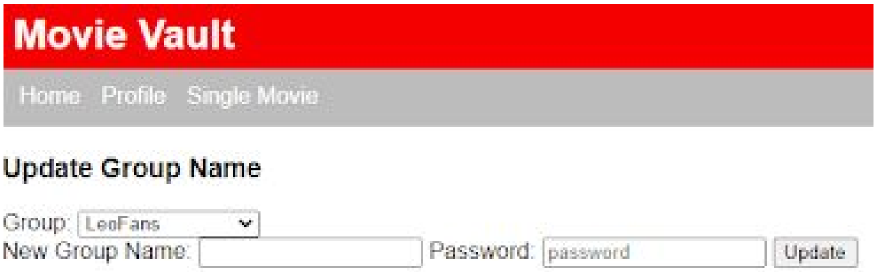
Group Name:

Email: simaxon98@gmail.com

Queries:

1. "SELECT email FROM Member INNER JOIN Account ON Member.account_ID WHERE Member.account_ID = ?"
2. "INSERT INTO Groups (groupName) VALUES (?)"
3. "INSERT INTO Apart_Of (groupID, email) VALUES ((SELECT groupID FROM Groups WHERE groupName = ?),?)"

13.) READ/UPDATE GROUPS



Update Group Name

Group: LeoFans

New Group Name:

Password: password

Queries:

1. "SELECT groupID as id, groupName From Groups"
2. "UPDATE Groups SET groupName = ? WHERE groupID = ?"

F. Triggers/Procedures/Functions

Triggers:

Overall Score Critic Insert: After a rating entry is inserted into the Rating Entry table, this trigger will update the overall critic score of the movie the rating entry is for.

Overall Score Audience Insert: After a rating entry is inserted into the Rating Entry table, this trigger will update the overall audience score of the movie the rating entry is for.

Overall Score Critic Update: After a rating entry is updated in the Rating Entry table, this trigger will update the overall critic score of the movie the rating entry is for.

Overall Score Audience Update: After a rating entry is updated in the Rating Entry table, this trigger will update the overall audience score of the movie the rating entry is for.

Overall Score Critic Delete: After a rating entry is deleted from the Rating Entry table, this trigger will update the overall critic score of the movie the rating entry is for.

Overall Score Audience Delete: After a rating entry is deleted from the Rating Entry table, this trigger will update the overall audience score of the movie the rating entry is for.

Procedures:

Create Account: This procedure takes in an email, name, company, username, and password. When this procedure is called, it first creates an account with the given username and password. Then, it creates a member using the email, name and company and then assigns the member to the accountID of the account that was just created.

Functions:

Get Average Critic Score: This function takes in a movieID. When this function is called, it finds all the rating entries with the same movieID and with an account where the company field IS NOT NULL. It then finds the average of those rating entries scores.

Get Average Audience Score: This function takes in a movieID. When this function is called, it finds all the rating entries with the same movieID and with an account where the company field IS NULL. It then finds the average of those rating entries scores.