Movie Rank
Milestone 1
Semantic Model

CSC 675-03
Team 02

Zac Henney
Ahmad Rangeen
Bingcheng Wu
Sandesh Basnet
Tharmika Thiyagarajah
Tuan Le

Table of Contents

Executive Summary

Our application, MovieRank, will allow our users to be able to search up movies, view movie ratings and provide their own ratings of the movie, if they chose to. The importance of this application is that users are seeing ratings of movies that were posted by regular people. Users don't have to just rely on what the critics rate the movies as sometimes a critic's point of view is much different than the general audience. Critics and the general audience do not always see eye to eye when it comes to rating a movie. MovieRank looks to provide audiences with ratings based on people like them.

MovieRank is designed to make it easier for users to search movies and only movies and decide if its worth watching depending on whether they are satisfied with the ratings or not. Once users finally decide on a movie to watch, users can watch it, come back to the website and give their own rating so other people can use it to decide if they want to watch that same movie. In order to do all this, users will have to create a user profile and be logged in.

MovieRank's Database architecture will be set up in a way where the entities Actors and Movies will require total participation from each other. As stated in the second paragraph, MovieRank is only for movies and nothing else, not even tv shows. To generate a movie's score, advanced SQL queries will be used to average all user ratings. This will allow the rating to be updated live as new ratings come in.

Entity Descriptions

User: A user, who is registered with the site and logged in, can browse movie listings, give movies a rating, and view the rating and information of movies.

Movie: The entity that represents further details about the movie. It contains information about the name, genre, details about the actor, release date, run time, producers info, and rating.

Genre: This entity provides a genre that can be applied to each movie. Examples of genres include “Comedy”, “Action”, and “Horror”.

Actor: The entity that contains details about the actor such as their name, age, and the portrait of the actor.

Rating: The ratings entity will be used to store the ratings given to each movie by each user . Ratings only accepts ratings from the user, who is registered with the site.

Studio: This entity is used to store information about the movie studios involved in producing the movies found in the movie entity. The studio entity contains information such at the name of the studio and the date it was founded.

Business Rules

This section contains the current business rules for the Movie Rank database application.

Actor
1. For an actor to exist they must star in at least one movie

Genre
2. A genre must be used to categorize at least one movie

Movie
3. A movie can only be categorized as one genre
4. For a movie to exist it must have at least one actor starring in it
5. A movie must be produced by only one studio
6. A movie can receive multiple ratings from multiple users

Rating
7. Each movie rating given by a user can only belong to one single user

Director
8. For a director to exist they must have directed at least one movie

User
9. A user can only input one rating at a time.

Functional Requirements

This section includes our proposed list of entities, their relations, and attributes. Unless noted, entity attributes are single valued and atomic.

1. Actor (weak)
   Relations: Stars In
   Attributes:
   - aid (key)
   - name
   - dob
   - age (derived)
   - thumbnail_url

2. Genre (weak)
   Relations: Categorized As
   Attributes:
   - gid (key)
   - name

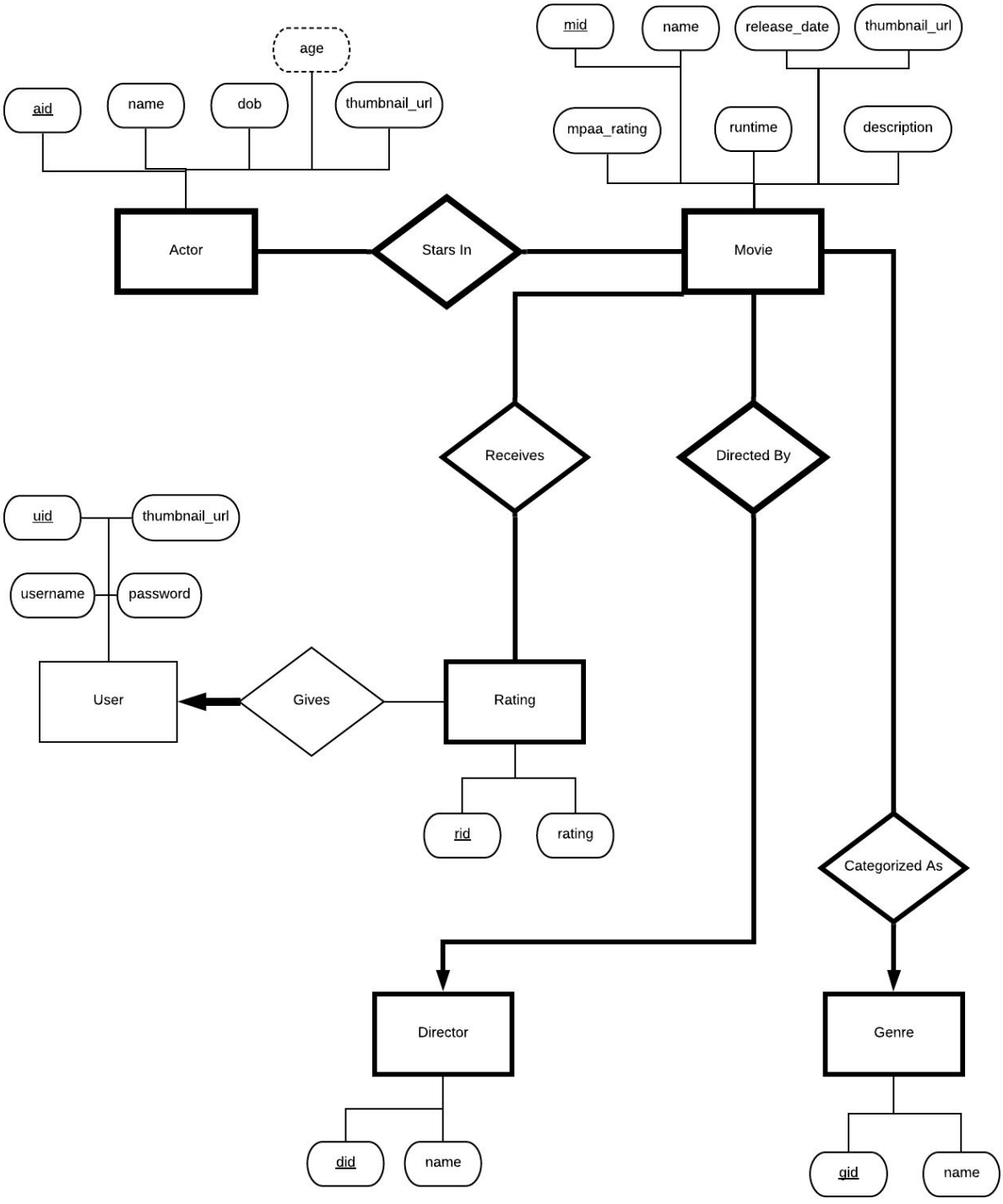
3. Movie (weak)
   Relations: Categorized As, Produced By, Receives, Stars In
   Attributes:
   - mid (key)
   - name
   - release_date
   - thumbnail_url
   - mpaa_rating
   - runtime
   - description

4. Rating (weak)
   Relations: Gives, Receives
   Attributes:
   - rid (key)
   - rating

5. Director (weak)
   Relations: Produced By
   Attributes:
      - did (key)
      - name

6. User (strong)
   Relations: Gives
   Attributes:
      - uid (key)
      - username
      - password
      - thumbnail_url

Entity Relationship Diagrams



ERD Tests

Business Rule #	Entity 1	Relationship	Type Rel	Entity 2	Pass/Fail	Modify
1	Actor	Stars In	M:M	Movie	Pass	
2	Genre	Categorized As	1:M	Movie	Pass	
3	Movie	Categorized As	M:1	Genre	Pass	
4	Movie	Stars In	M:M	Actor	Pass	
5	Movie	Directed by	M:1	Director	Pass	
6	Movie	Received	1:1	Rating	Fail	Change to M:M
7	Movie	Received	M:M	Rating	Pass	
8	Rating	Gives	M:1	User	Pass	
9	Director	Directed by	1:M	Movie	Pass	
10	User	Gives	1:M	Rating	Pass	

Non-Functional Requirements

1. Application shall be developed, tested, and deployed using the tech stack, and server approved by the class CTO/CEO.

2. Application shall be optimized for the recent two versions of standard desktop/laptop web browsers such as Firefox 65.0.1 and Chrome 73.0.3683.103.

3. Data shall be stored using Postgres DBMS.

4. Application will be deployed on Heroku

5. Application shall support more than 50 concurrent users at any time.

6. Privacy of the users shall be protected through encryption of data, and all privacy policies should be communicated to the user.

7. Application shall be developed using best UX practices (if time permits).

8. Application shall be able to support storage of ~ 500 MB of data.

Contributions

Zac Henney (team lead) - 10
Ahmad Rangeen (github master) - 10
Bingcheng Wu - 8
Sandesh Basnet - 7
Tharmika Thiyagarajah - 8
Tuan Le - 8