CS5200 Project – Moviestop

Under Prof. Ghita Amor-Tijani

College of Computer and Information Science

Northeastern University

GROUP 8: DUBEYJOLAKULAMAHESHAN

Sumeet Dubey

And

Shreyank Jolakula Maheshan

Table of Contents

Table of figures

Section 1: README

Group number: 8
Group name: DubeyJolakulamaheshan
Group members: Sumeet Dubey, Shreyank Jolakula Maheshan

We have included the admin functionality in this submission which was not part of our presentation.

Files Included:

1. All webpage files with client side javascript and server functions are in the app directory
2. Self contained sql file of database along with any procedures, functions and views.
3. Original movies and awards tables used as data.
4. Few database backups throughout the course of this project

Software and libraries used:

1. WebStorm IDE (for node app)
2. MySQL workbench (for database)

Instructions to run:
The project is written on nodeJS and angularJS. All the files to run the webpage is in the app folder. A self contained sql script of our database is present in the db folder, along with some other dumps we generated over the course of this project. To run the website you will need to install Nodejs from here. We have made it on version 6.0, so any 6.x version should work. After installing node, you can navigate to the directory 'app' and run the command npm update followed by npm install to install all the necessary node packages.
To interact with the db, you will need to load our database into your local workbench instance and provide the username and password in the server.js file. After this run node server.js and the terminal should display a message 'Connected!' The website can be opened by navigating to localhost:3000

Use cases:

- User can view movies or register.
- User registers by providing username, password and email address.
- User can enter and save addition profile info or a new payment method.
- User can search for movies to purchase or search for academy award winners.
- A search for movie purchase can be done using name of the actor, director, or the movie.
- User can view the movies which are available for purchase and select.
- User selects the quality of the movie they want to buy, if multiple available.
- User selects one of the saved payment methods or adds a new one.

- User confirms purchase and can see the newly purchased movie in their ‘my movies’ tab.
- User can search academy awards by winner or award description

Section 2: Technical Specifications

Software, Apps and Languages that were used to develop the project:

Front-end: AngularJS
Back-end: NodeJS
Database: MySQL 5.7
Software: MySQL Workbench

Section 3: UML/EER Diagram

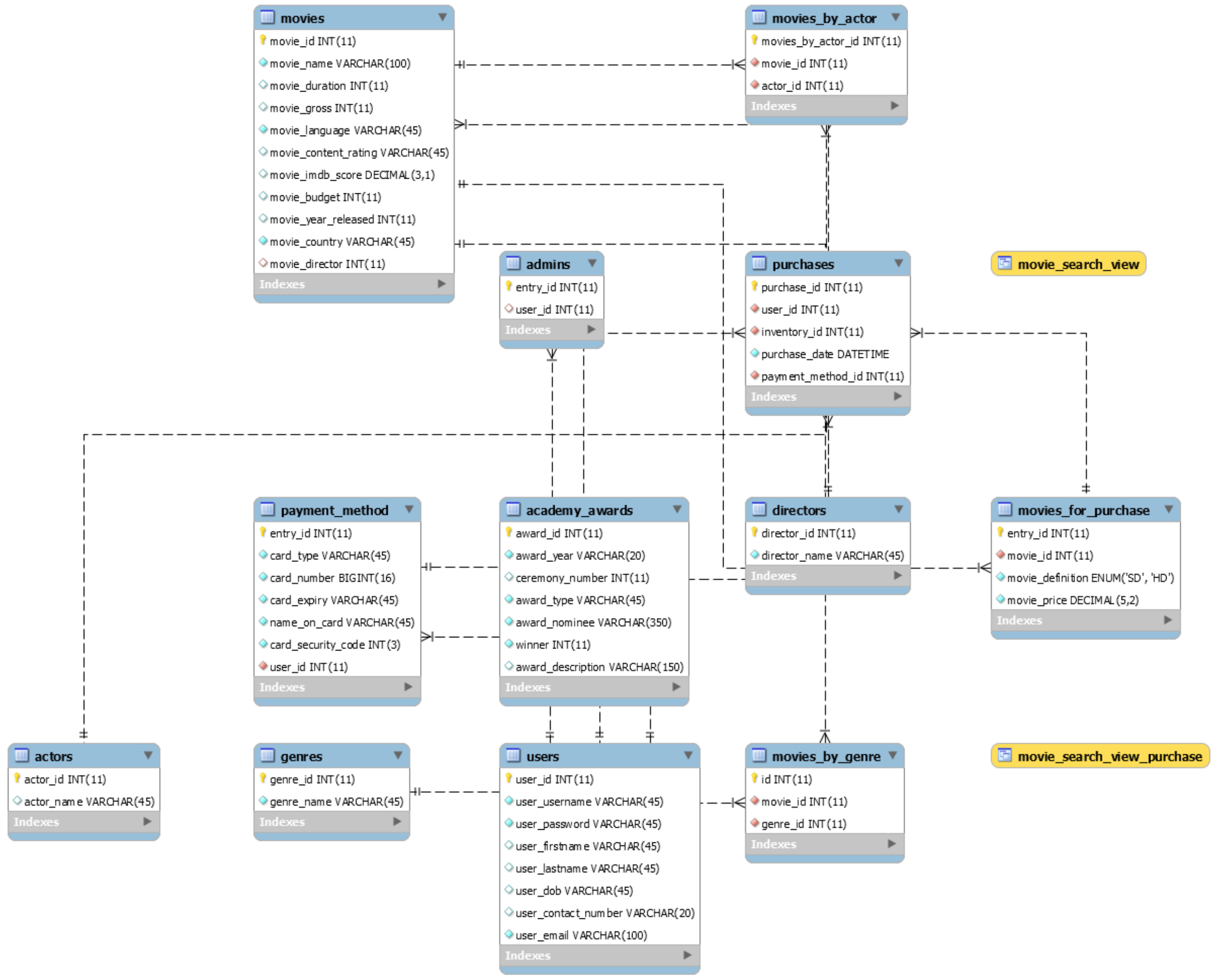

Figure 1: UML/ER Diagram

The data which we started with had few redundancies. The fields were not segregated. To remove these redundancies and hence to reduce the overhead on db communication, we normalized the tables. The table ‘movies’ had attributes which contained directors’ and actors’ data (three columns in case of actors). We moved these out into 2 new tables, directors and actors.

It is possible for a single actor to have acted in multiple movies and multiple movies to have the same actor present in them. We therefore separated out a relational table movies_by_actors that references movie_id and actor_id as foreign keys. We normalized it by making a separate table for ‘actors’ and used actor_id as the primary key. We have assumed each movie has one director, therefore director_id is included as a foreign key in the movies table. We similarly separated the genres tables by adding a relational table movies_by_genre.

Section 4: User Flow diagram

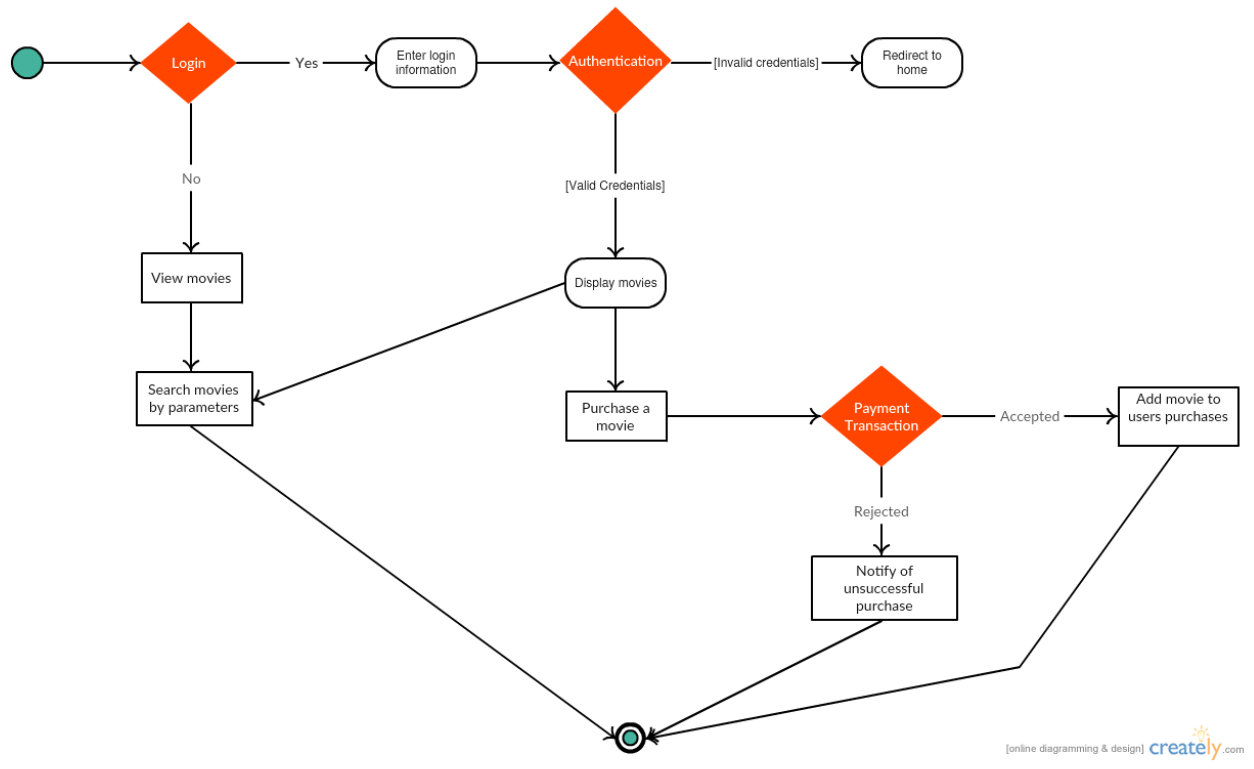

Figure 2:User flow Diagram

Step by step user interaction:

1. User creates an account by creating a username and password.
2. Registered user searches for movies by entering the name of the name.
3. Movies can be searched using different parameters such as actor, director.
4. The academy awards won can be searched using movie, cast etc. This is done by entering the details in the provided fields.
5. A set of movies are available for sale. Registered users can buy digital copies of these movies. Each movie is available in SD or HD (or both).
6. Registered user will select one of Standard-Definition or High-definition and proceed to payment.

7. Registered user selects one of the existing payment methods or creates a new payment method. The payment information such as card number, security code, name and expiry date are entered in the provided fields
8. Upon authorization of payment, purchase is added to the users purchased movies list.

Interactions for admin:

1. Admin can log in by select the 'login as admin' option on the login page.
2. If the credentials provided are correct, the user is taken to the admin page.
3. Admin can insert, add, update or delete a user, a movie, or a movie available for purchase.

Section 5: Lessons Learned

1. Technical expertise gained:
   Over the course of the project, we learnt to use the MySQL Workbench effectively. We learnt to write procedures, functions and queries in MySQL. The tasks and milestones for the project equipped us with sufficient knowledge to design and create a database. We also learnt to connect the front-end with database and effectively handle errors.

2. Time management and domain insights:
   The regular milestones such as project proposal, project progress report and presentation always kept us on our toes. We did not face any problems with the time management and data domain. A few tasks in data normalizing took some time.

3. Realized and contemplated approaches:
   When we first selected this domain, we had planned to focus only on the academy awards part of the project. But when we moved forward with the project, we started including more functionalities such as purchase of movies, inventory of available movies, admin panel etc.

4. Document any code not working in this section:
   The code is working properly.
   Some of the pages might take a few seconds to loads because of huge tables. We have tried to reduce this by loading as per the user scrolls on the page.
   Some of the columns in the admin update tables are greyed out, like the movie name in the movies for purchase table. This is because the movie name belongs to a separate table 'all movies' and it should be updated from that table (along with the other attributes for a movie). These changes will be automatically reflected in the movies for purchase table.

Section 6: Future work

The database can be used to include other tables such as user reviews and critic reviews. These reviews can be used to give a rating to the movie. This can probably be done using weighted averages with more weightage to critic reviews.
We can include the functionality of suggesting the movies to a user based on his previous purchases. This can be done by matching the genres of the movies he has already bought with the other movies in the database. We can also add the functionality of renting a movie for a limited duration.

Section 7: Procedures, Functions, Views, Indexes, Error handling

1. add_user()
   This procedure takes the username, user email, phone number, first and last name as input and adds the user to the user table.

2. add_purchase()
   This procedure takes the movie_id, payment_method, user_id, definition as input and then adds the movie to the user's purchase list.

3. add_movie_for_purchase()
   This procedure takes the movie_id, price, definition as input and adds the movie to 'movies_for_purchase' table.

4. add_movie()
   This procedure takes the movie_id, movie_name, director_id, year of release, imdb_rating, gross as input and adds the movie to 'movies' table.

5. by_actor()
   This procedure takes the actor's name as input and returns the movie names and year of the movies that he has acted in.

6. by_actor_by_movie()
   This procedure takes the actor's name or movie name as input and returns the academy awards associated with the input.

7. by_director()
   This procedure takes the name of the director as input and returns the movies directed by him.

8. get_user_movies()

This procedure takes the user_id as input and returns all the movies purchased by that particular user.

9. movies_available()
   This procedure displays the movies which are available for purchase along with their price and quality.

10. views_all_movies_p()
    This procedure displays the view movies_search_view_purchase.

11. view_all_movies()
    This procedure displays the view movie_search_purchase.

12. update_movie_for_purchase()
    This procedure takes the movie_id, price, quality as input and updates the record in 'movies_for_purchase' table.

13. update_movie()
    This procedure takes the movie_id, movie_name, director_id, year of release, imdb_rating, gross as input and updates the movie in 'movies' table.

14. search_by_movie(), search_by_actor(), search_by_director().
    These procedures take the movie's name, actor's, director's name as input respectively and display the matches in the view 'movies_search_view'

15. payment_info()
    This procedure takes the user id as input and displays the payment_information of that particular user.

16. view_awards()
    This procedure displays the list of award winners and the movies that they got the award for.

17. get_inventory_id()
    This function takes the movie id and movie definition as input and selects the entry_id.

18. check_admin()
    This function takes the user id as input and checks if the user is an admin or not.

19. user_table()
    This function takes username, email, first and last name, contact_number, password and inserts the user into the user_table.

20. movie_search_view
    This is a view which displays the details of the movies in the database.

21. movie_search_view_purchase
    This is a view which displays the details of the movies which are available for purchase.

    Secondary indexes are included in various tables (like the username and email in users table). Unique constraints have been put where needed (combination of movie_id and movie_definition should be unique in movies_for_purchase table)
    The error handling in cases such as no input specified is done on front end for efficiency. The other errors and checks like not allowing the user to create an account with an already existing username is done on the backend. We have tried to report most of the errors given by mysql to the frontend, especially for admin functions.