DSCI 351 - Spring 2024 Project
Fan Zhang, Binqian Chai, Annan Chen
Prof. Wu
May 5, 2024

Movie Management System

Introduction

The “Movie Management System” is a dynamic application designed to enhance the movie-watching experience. Our management system includes two parts– a movie recommendation system and a movie rating system. Before watching the movie, users can use the movie recommendation system to search movies from our global movie library and add, update, or delete movie attributes into the library based on their preferences. Users will receive tailored recommendations based on their genre preferences. After watching the movie, users can use the movie rating system to search the movie they just watched for movie reviewing, add or delete movies that they would like to rate, and update “inappropriate” rating numbers.

Planned Implementation

Our application integrates two NoSQL databases to store data and do the functionalities and uses Streamlit to design a Web browser-based GUI.

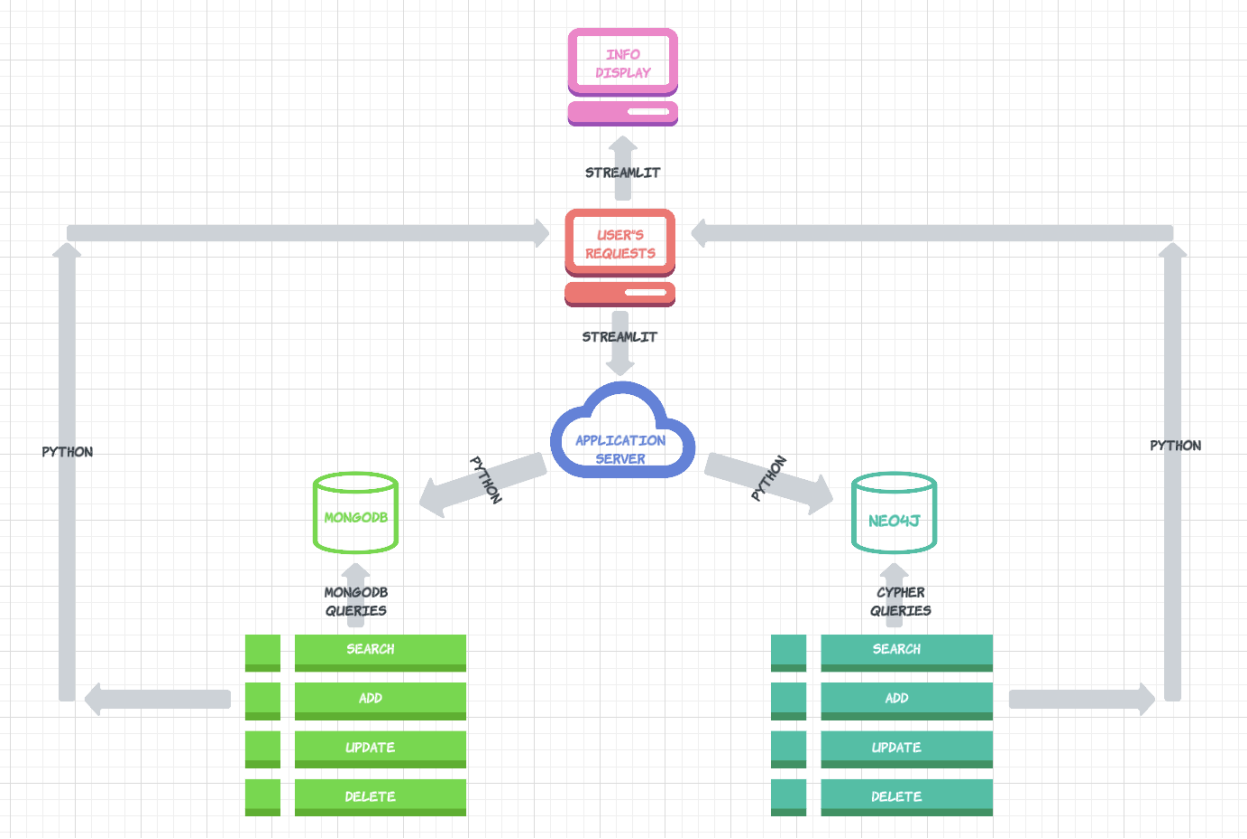
1. MongoDB (Movie Recommendation System)
    - Database Design: Design the MongoDB schema to store the global movie dataset. This includes defining the structure of documents for movies, which will have attributes like title, director, release year, duration, country, rating, genre, and description.
    - CRUD Operations: Implement the functionality to search movies based on specific genre, add new movies to the stored dataset, update movie attributes, and delete existing movies from the dataset.
2. Neo4j (Movie Rating System)
    - Database Design: Design two nodes for movies and directors to store the related attributes, and a relationship that connects the movie nodes with director nodes. This structure supports complex queries to generate personalized movie ratings.

- CRUD Operations: Implement the functionality to search movies, add new movies to the stored dataset, update movie ratings, and delete existing movies from the dataset.

3. Streamlit (Web browser-based GUI)
    - Developing a user-interacted interface.
    - Connecting the movie recommendation system and the movie rating system into one web-browser.

Architecture Design

1. Flow Diagram

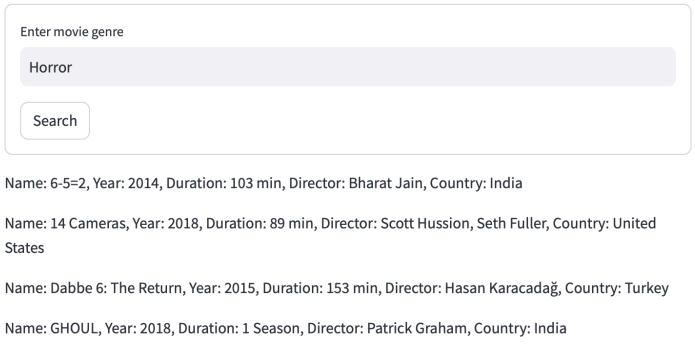


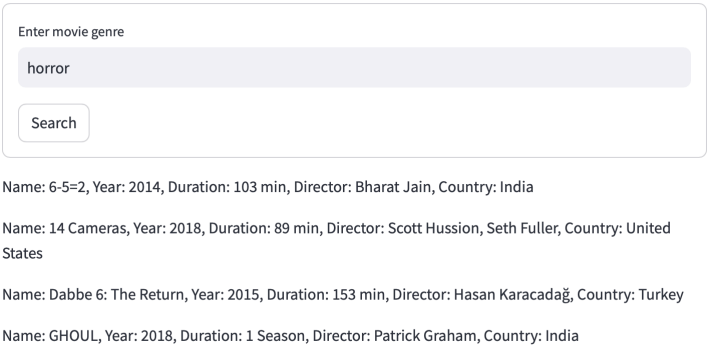
2. Description
    - User Interaction: The user accesses the Streamlit interface to perform actions like searching, inserting, modifying, or deleting.
    - Database Operations: The application server processes the user's request. For example, a search request goes through the search_movies function, which constructs a MongoDB query and retrieves the relevant data.

- Data Retrieval and Display: The results from MongoDB or Neo4j are returned to the application server and subsequently displayed to the user through the Streamlit UI.

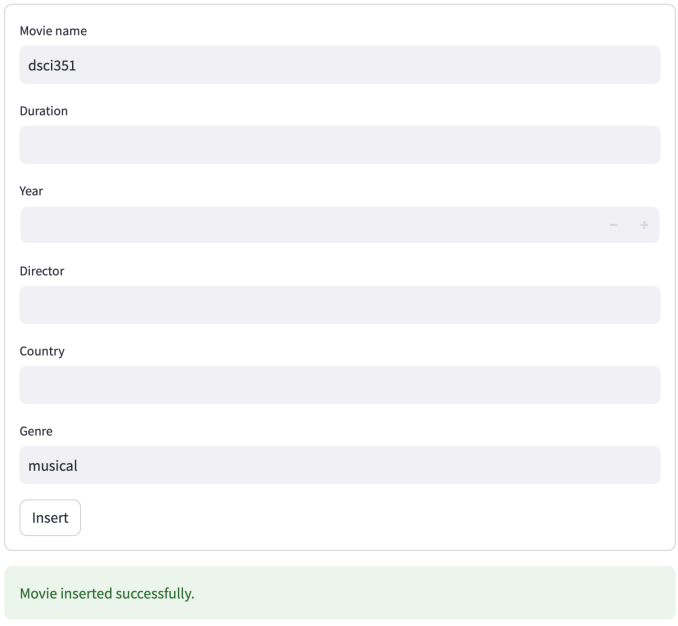
Implementation

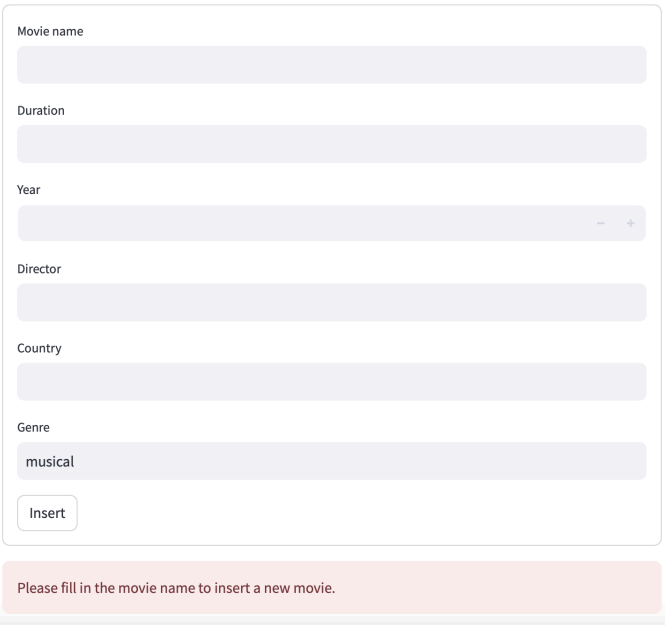
1. MongoDB (Movie Recommendation System): A separate load_mongo.py file is first used to load the movie data into MongoDB database. Python coding is implemented to connect the web-browser Streamlit GUI to the MongoDB queries.
    - Search: The user can search movies by typing at least one key word of genre. The system will display all movies under the user-entered genre in our movie library with name, year, duration, director, and country. The genre entered is case insensitive.

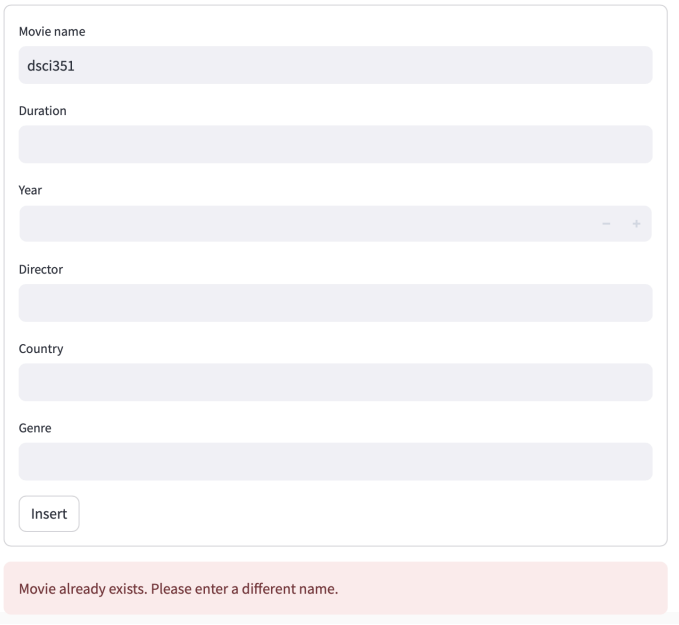




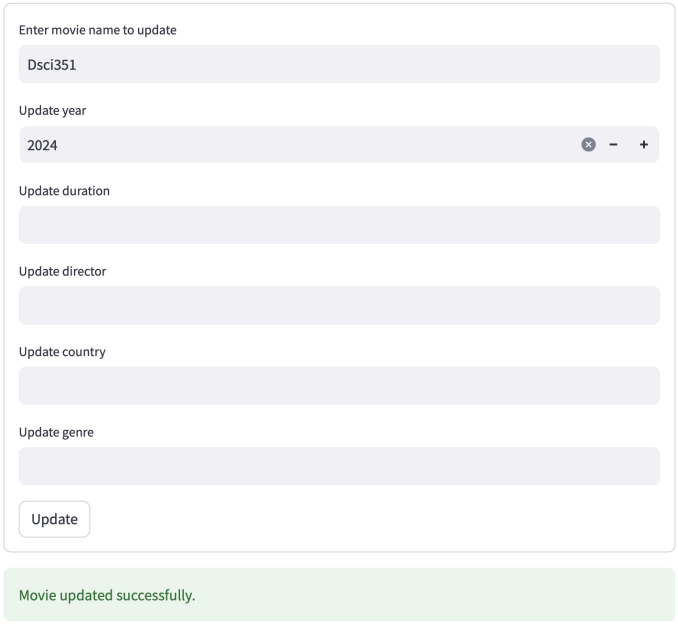
- Insert: The user can insert new movies by using the inserting section. A movie name must be provided by the user, while other attributes (duration, year, director, country, and genre) can be left blank. If the user left the movie name blank, our system will have an error message asking the user to fill in the movie name to insert a new movie. The user is not allowed to add a movie that has the same name as the one in our movie library because our system will use the movie name as the primary key to search from the movie library to complete modifying or deleting tasks. If the user enters a movie name that is already in our movie library, the system will ask the user to enter a different name. The checking step in our system is set to be case insensitive. If each of the above conditions is met, the system will print a success message.

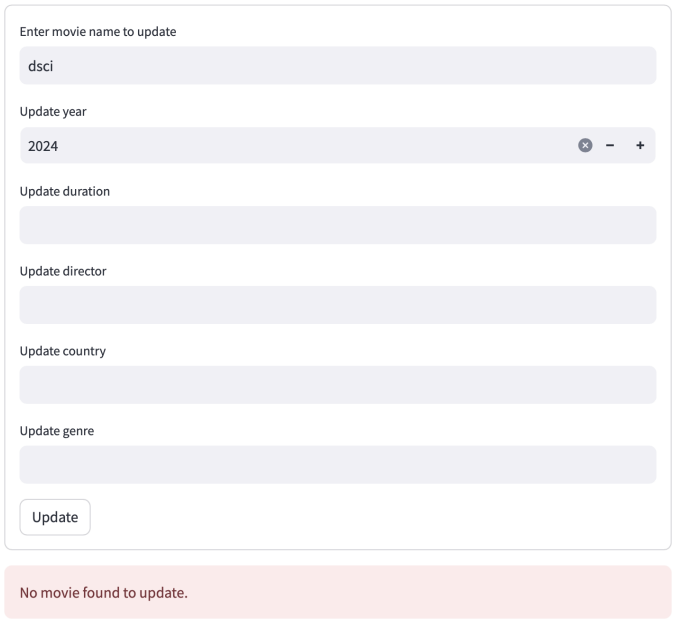




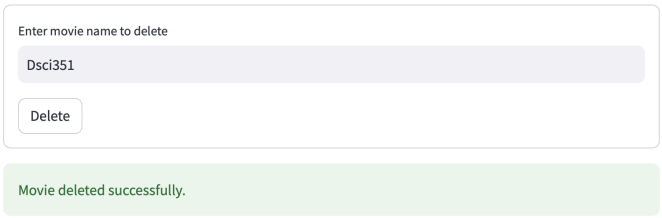


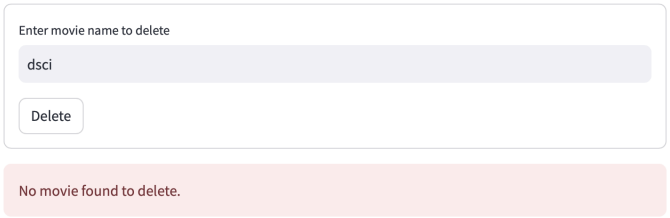
- Modify: The user can update the attributes (year, duration, director, country, and genre) of the movie by entering the movie name. The movie name is set to be case insensitive, meaning that the system will update the attributes of the movie based on the name matching, no matter if the user uses upper or lower cases. If the movie attributes are updated, a success message will be given. If the movie the user wants to update is not in our system, an error message will be given.





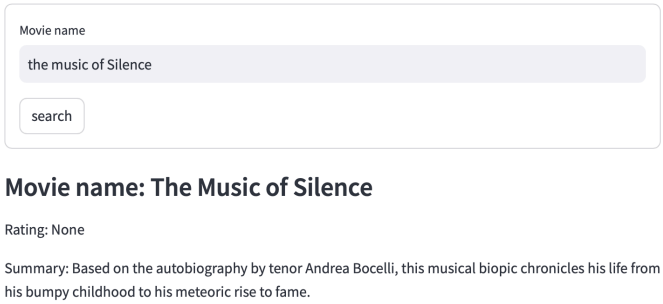
- Delete: The user can delete the movie from our movie library using the deleting section. The user can enter a movie name without worrying about the upper or lower cases to delete a movie since our system is set to be case insensitive. If the movie is deleted, a success message will be given. If the entered movie name is not in the movie library, an error message will be given.

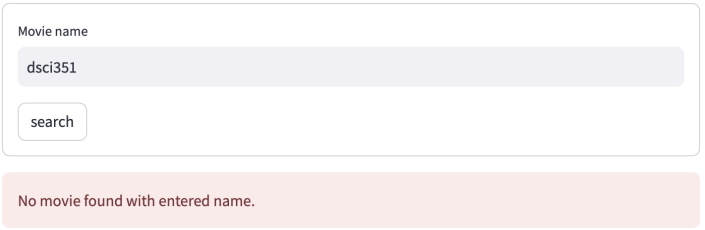




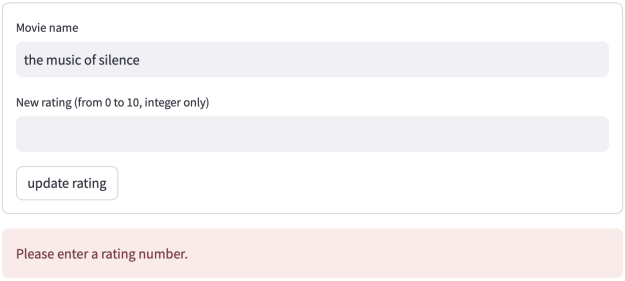
2. Neo4j (Movie Rating System): The same dataset is used for the movie rating system. But the dataset is manually separated into three csv files– a movie file containing the attributes of title, country, release year, duration, rating, genre, and description, a director file containing the director name, and a relationship file matching the movie title with the director name. The three files are loaded into the Neo4j desktop using the cypher query language. Python coding is implemented to connect the web-browser Streamlit GUI to the Neo4j queries.
    - Search: The user can search the movie by entering the movie name. The movie name being searched is set to be case insensitive. If the entered movie name is in

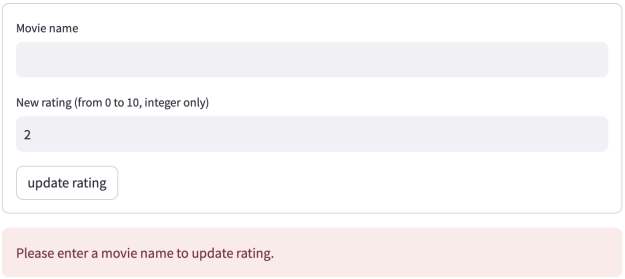
our movie library, the system will give out a brief summary of the movie and its current rating. The user can read through the summary to check whether the movie is the one they watched and would like to check, give or update the rating. If the current rating is null, the rating will be shown as “None”. If no movie name is found in our movie library, the system will print out an error message.

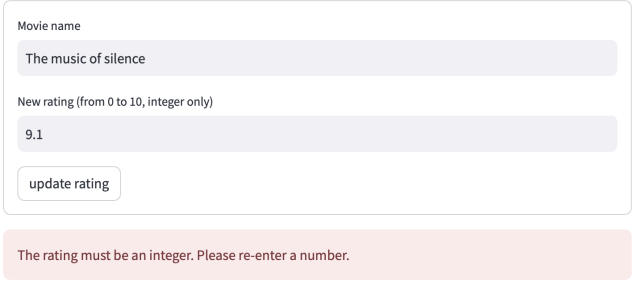


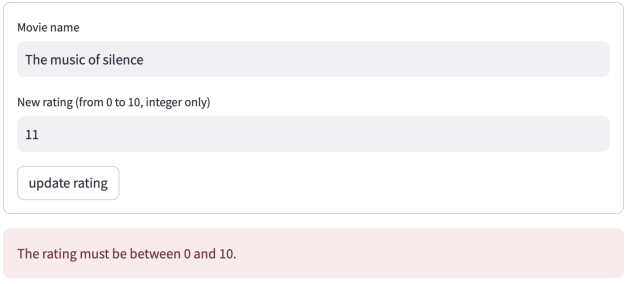


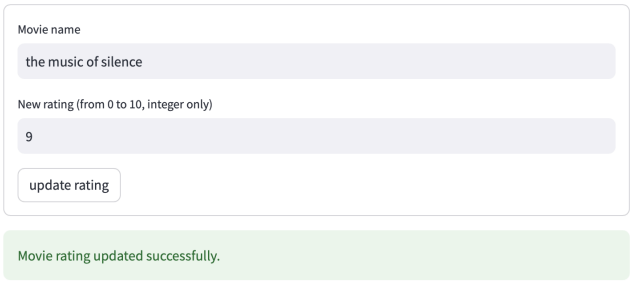
- Modify: The user can update the current rating that he/she is not satisfied with using the updating section. The movie name and new rating number must be entered, otherwise the system will print out an error message. The movie name is set to be case insensitive. The new rating number is set to be an integer between 0 and 10 (inclusive). If the entered new rating doesn’t meet the conditions, an error message will be displayed. It is fine to enter a new rating number that is the same as the previous one. A success message will still be displayed. If the movie entered is not in our movie library, an error message will be printed.

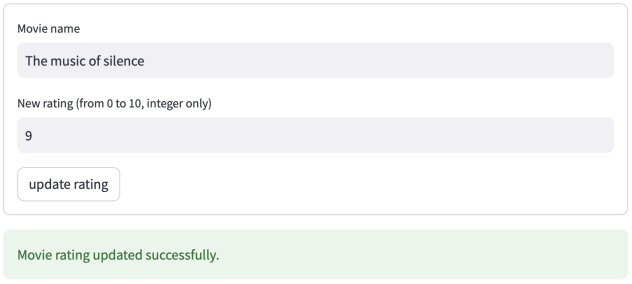


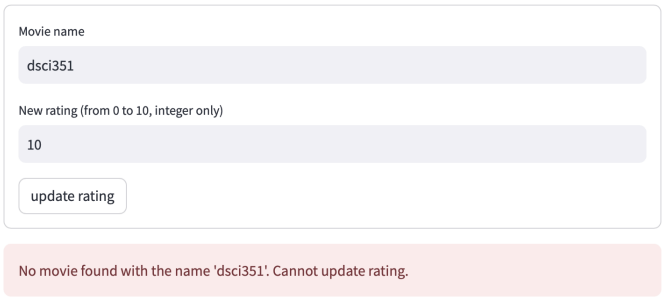




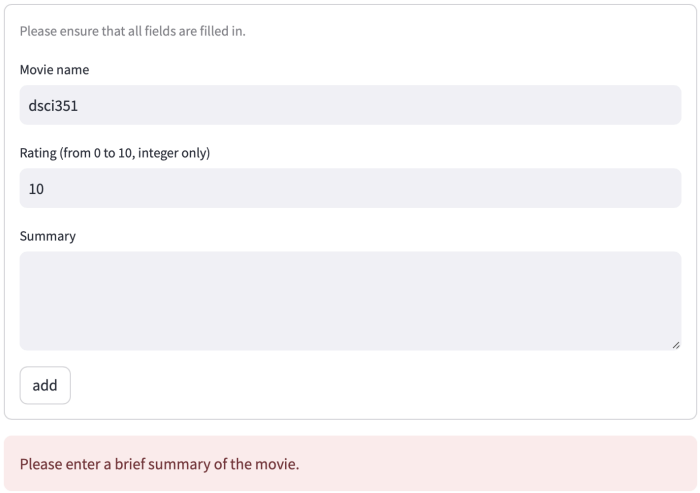


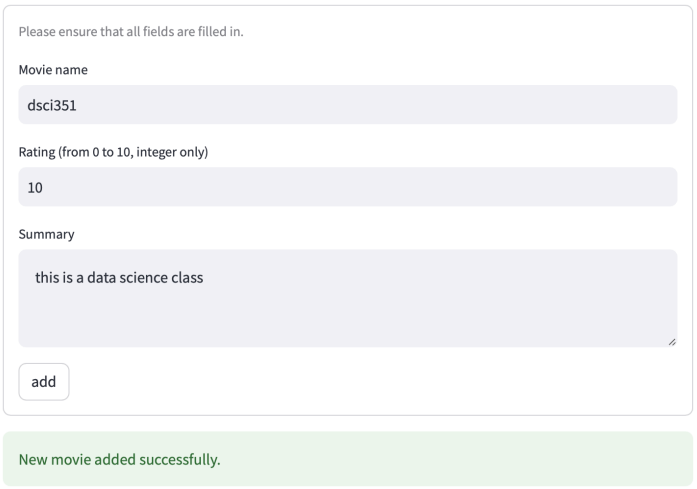




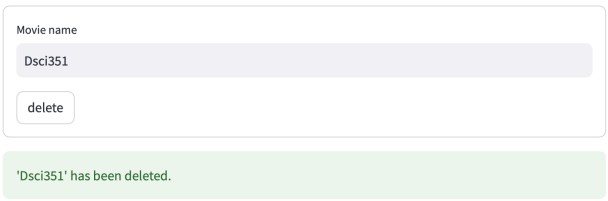


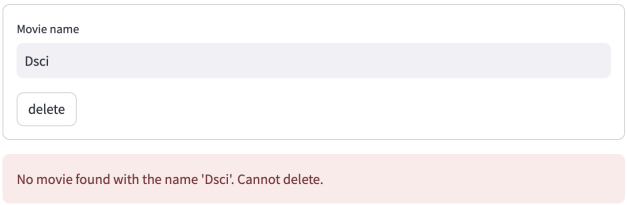
- Insert: The user can add new movies that are not in our movie library but would like to give a rating to it using the adding section. The blank fields must all be filled, otherwise an error message will be displayed. The rating has the same rule as the one designed in the updating section: the entered number must be an integer between 0 and 10 (inclusive). If the movie name entered is already in our movie library, an error message will be printed out. The movie name is set to be case insensitive. If the adding is successful, a success message will be displayed.





- Delete: The user can delete movies from our movie rating system library by entering the movie name. The movie name is set to be case insensitive. If no movie is found in our movie library, an error message will be displayed. If the movie entered is deleted successfully, a success message will be given.





Challenges Encountered

1. We had a hard time deciding what kind of tasks each of the two databases could do because we were considering integrating the two databases into one system but each doing the similar search, insert, modify and delete tasks and using the same dataset. We finally decided to have two subsystems in the movie management system and implemented MongoDB to manage the movie recommendation system tasks and Neo4j to manage the movie rating system tasks. By refining the project ideas, the final appearance of our project is slightly different from the one in our project proposal.
2. All of our group members are new to web-browser GUI design, so we took a long time to figure out the tool that is the most “friendly” for us to use to do the user interface design.

3. We were thinking about whether to include more data in our movie library so that the user could have a more thorough and diversified movie suggestion. But ensuring the quality and completeness of the dataset can be challenging.

Individual Contribution

1. Fan Zhang
    - Data Preparation: Found a netflix film dataset from Kaggle as the dataset of our movie management system. The dataset originally had more than 5000 rows of data, including columns of film ID, title, director, cast, country, data added date, release year, netflix rating, film duration, type of movie, listed_in (keywords describing the genre of the movie), and a short description about the content of the movie. Did a data cleaning process to filter out those null data and made the dataset to have 300 rows with eight selected columns. Prepared three separated files for data loading in Neo4j.
    - Web-browser GUI Design: Used streamlit to connect the two databases to the front end for user interaction.
2. Binqian Chai
    - Database Design (Neo4j): Responsible for loading the dataset into the Neo4j desktop and developing functionalities using cypher queries integrated with python coding for users to do search, add, update, and delete tasks in our movie rating system.
3. Annan Chen
    - Database Design (MongoDB): Responsible for loading the dataset into the MongoDB database and developing functionalities using MongoDB queries integrated with python coding for users to do search, add, update, and delete tasks in our movie recommendation system.

Conclusion

Our “Movie Management System” combines the functionalities of a movie recommendation system and a movie rating system, utilizing two NoSQL databases—MongoDB and Neo4j—to handle data storage and retrieval tasks. The movie recommendation system improved user engagement by providing personalized movie suggestions based on individual preferences of genre. The movie rating system facilitated user interaction by enabling users to update and rate

movies effectively. The users can perform CRUD operations such as searching, inserting, updating, and deleting movie entries in both of these two subsystems. The design of our application also reflects the characteristics of NoSQL databases. Both MongoDB and Neo4j support horizontal scalability, which is beneficial for web applications expecting to grow in data volume. The schema flexibility is shown by our application. MongoDB allowed for easy modifications and extensions of the movie data without the need for extensive database refactoring. Neo4j manages complex relationships between data entities efficiently. In conclusion, our project experience will help us in our future endeavors in the field of data science and application development.

Code Submission Link

https://drive.google.com/drive/folders/17tS9F3WgEjn6unUMkmbm0Q5d7uzwcqm8?usp=sharing