CMPUT 291 - Mini Project 2

DESIGN DOCUMENT

Submission Date: March 30th, 2022

Group Members:

1. Hridyansh Dugar , ccid:dugar (1667524)
2. Swastik Sharma, ccid: swastik (1670432)
3. Rana Thind , ccid:ranasunj (1670185)

---

1. **A general overview of our system with a small user guide:**

The goal of this project is to learn the concept of working with data stored in files and NoSQL databases which is done by building and operating on a document store, using MongoDB. In this project, a software system is built to keep the enterprise data in a database and to provide services to users. It will store data in a database.The user can create the database by running the tsv-2-json.py and then load-json.py after starting the MongoDB server. A user can run the program using the terminal and typing in “python3 operations.py” and interact with the interface to perform functions such as searching for titles,genres or for cast/crew members and can add a movie and cast/crew member as well. After each action, the user is able to return to the main menu for further operations and can end the program if he/she wishes to.

User Guide:

- The user starts the application by making sure that a MongoDB server is running and the database is populated
- Then the user opens a terminal window and enters the command - python3 operations.py
- After which the user can use the various operations.

2. **Detailed Design of the program:**

Phase 1: Building a document store:

Here we implemented two programs. One program will read the four tsv files in the current directory and convert them to json files. The other program will take those four json files in the current directory and construct a MongoDB collection for each.

Phase 2: Operating on the document store:

1.) ***Search for titles :-*** The user is able to provide one or more keywords, and the system retrieves all titles that match all those keywords (AND semantics). A keyword matches if it appears in the primaryTitle field. A keyword also matches if it has the same value as the year field. For each matching title, display all the fields in title_basics. The user should be able to select a title to see the rating, the number of votes, the names of cast/crew members and their characters (if any).
2.) ***Search for genres:-*** The user is able to provide a genre and a minimum vote count and see all titles under the provided genre that have the given number of votes or more. The result is sorted based on the average rating with the highest rating on top.
3.) ***Search for cast/crew members:-*** The user is able to provide a cast/crew member name and see all professions of the member and for each title the member had a job, the primary title, the job and character (if any). Matching of the member name should be case-insensitive.
4.) ***Add a movie:-*** The user is able to add a row to title_basics by providing a unique id, a title, a start year, a running time and a list of genres. Both the primary title and the original title are set to the provided title, the title type is set to movie and isAdult and endYear are set to Null (denoted as \N).
5.) ***Add a cast/crew member:-*** The user is able to add a row to title_principals by providing a cast/crew member id, a title id, and a category. The provided title and person ids should exist in name_basics and title_basics respectively (otherwise, proper messages is given), the ordering should be set to the largest ordering listed for the title plus one (or 1 if the title is not listed in title_principals) and any other field that is not provided (including job and characters) set to Null.

3. **Testing Strategies:**

- Started building the program by having all the set-up data in place and writing one query at a time. This allowed us to build a bug-free program in small steps.

- A small test database was also created to test the functionality of the queries.

- Error checks were performed by adding try and accept blocks.

- Rigorous error testing was done after every completed portion of the devised assignment goal.

4. **Group Break down Strategy:**

Break-down of the work items:

- Rana worked on Phase 2 which includes ‘Add a movie’ and ‘Add a cast/crew member’ and testing.
- Swastik worked on Phase 1: Building a document store and helped with the testing procedures.

- Hridyansh worked on the initial design of the Phase 2 which includes ‘Search for titles’ , ‘Search for genres’ and ‘search for cast/crew members’ , testing and error checking.

- At the end of each individual component we sat down together to integrate the components into the collective program.

Time spent of each member :

- Hridyansh spent approx.8-10 hours
- Swastik spent approx. 6-7 hours
- Rana spent approx. 5-6 hours

Method of coordination to keep the project on track

- The group met at the beginning of the project to discuss an overall strategy and divide the work among themselves.

- The group devised to use version control software such as Github

- The group also met on the submission day to go through the code together, divide it into separate files for improving code quality. Changes were also made in the code to ensure it works well with the python version on the lab machine

- The code was pushed on GitHub so the members could see the current state of the project at all times and review each others’ work as well.

- All the other communication was done in a group chat on a Discord server, including modifying work distribution as necessary and tracking progress.

- The design document was prepared by collaborating on a google doc