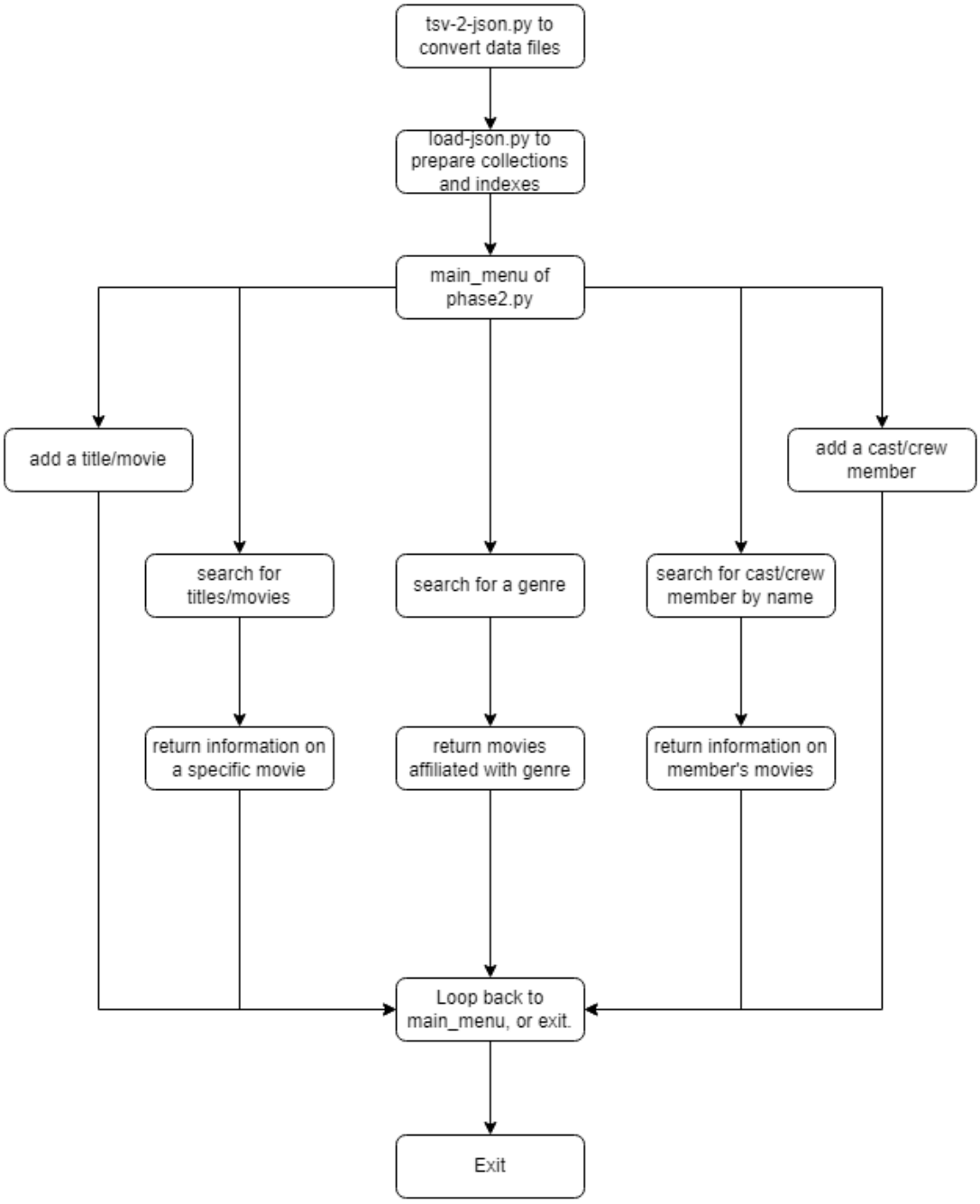
Report

Group members: Weiguo Jiang, Dylan Clarke, Liam Chen

(a) A general overview of your system with a small user guide

An illustration of program's workflow is given below:



The purpose of this project is to maintain a document store using pymongo and interaction with the MongoDB server. During the execution of the program, you will need to provide the port number of the server. After successfully connecting to the server, you will see the main menu providing the following operations: search for titles, search for genres, search for cast/crew members, add a movie, add a cast/crew member. You will return to the main menu after each operation and there's an option to exit the program.

(b) a detailed design of your software with a focus on the components required to deliver the major functions of your application
main_menu(): User interface providing the options stated in part (a) and allows graceful exit of the program

search_for_titles(): Users can provide keywords to search for a movie. The movies will be displayed if all keywords are in the "primaryTitle" field. The movies also will be displayed if one of the keywords appears in the "startYear" field. This is done by using full text index search. The user can then select a movie from the displayed list, then ratings, number of votes, cast/crew member, and their characters will be displayed. This is using the left outer join, using the "lookup" function in MongoDB. (Note: these only will be displayed if the selection is within the boundary)

search_for_genres(): Users can provide a genre, then provide a vote count. The primary title and original title of the movies that have the given genre type and those that have the "numVote" greater than the provided vote count will be displayed. This is done by full text search on the "genres" field, and left outer join on the "tconst" field in order to retrieve the votes and ratings of the movies.
search_for_members(): The user should be 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). The result is shown for each individual title match instead of multiple titles under the same character.

add_a_movie(): The user is asked to individually input the required components of a movie for insertion into the title_basics collection. The ID of the movie must be unique, as to not violate any key constraints, and any attempt at inputting an unoriginal ID will be rejected, and the user will be prompted again. We assume that any type of input may be accepted for the title, including numbers, symbols, or other unusual characters. Both the year and runtime inputs are required to be numeric. Genres are accepted in the form of a list, with commas separating multiple genres that a user may choose to enter. After the user has entered in all these valid inputs, a new row will be inserted into title_basics with the user's values, complemented by null values for *isAdult* and *endYear*.

add_a_member(): To insert a cast or crew member into title_principals, the user is asked to input the id of an existing cast or crew member, and any IDs that are non-existent within name_basics shall be rejected. Similarly, the user must input an ID for an existing title, and if it does not exist within title_basics, the ID shall be rejected. The user may then input a category for the movie, and after which, the program will calculate the proper ordering for this new cast member. The maximum ordering currently for the title in title_principals will be found, and the ordering for this cast member will be set to that plus one. This includes the case of no existing ordering for that movie, where this cast member will simply become the first or "1". After this, the movie will be inserted into title_principals with complementary null values for job and character.

(c) your testing strategy

Testing is done by inputting legal values according to the specifications. However, error handling and case insensitivity are also implemented so users do not have to worry about making mistakes. After testing the functionality of usual inputs, we test edge cases extensively to find details we might have overlooked. Testing was performed to ensure that various combinations of actions resulted in intended behavior, whether it be searching within the initial database or testing data added within the program.

Assumptions:

1. We assume that all files can be loaded directly into main memory.
2. We assume the port number provided is valid, i.e., an integer.
3. We assume that if a new tconst is added to title_basics collection, it is not added to title_ratings collection. Hence searching by genre and vote counts will not display the title of newly added tconst.
4. We assume that the user sets up the database correctly using mongo before running the main program.

The program is tested on the lab machine and it runs successfully.

(d) your group work break-down strategy.
Weiguo Jiang: phase 1, main menu, task #3. 10hrs.
Liam Chen: task #1, task #2. 10hrs.
Dylan Clarke: task #4, task #5. 10hrs.
All of us participated equally in creating the Report.pdf