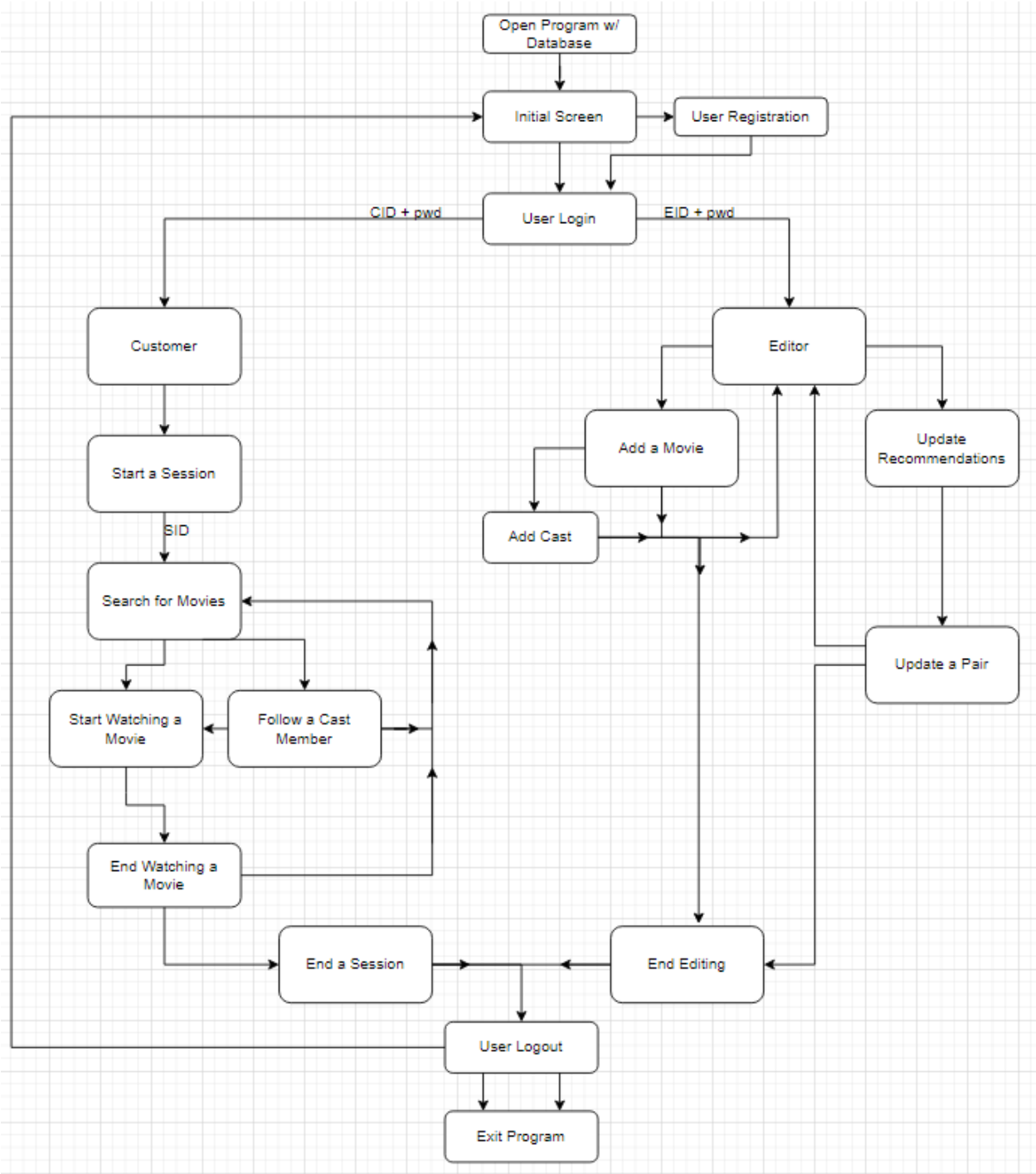
Report

Group members: Weiguo Jiang, Dylan Clarke, Liam Chen

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



The purpose of this project is to maintain a streaming service database. Customers using the service are allowed to open sessions, search for movies, watch movies and end sessions. Editors have the ability to add a movie to the existing movies and update recommendations.

Before login, we will first connect to a database. The database's path is provided via command line argument.
The program's interface is a login screen that asks the user to login as a customer or an editor and perform corresponding tasks. The respective tasks that each role can do is discussed above and will be discussed in greater detail in part (b).

(b) a detailed design of your software with a focus on the components required to deliver the major functions of your application

The **connect()** function has the same purpose as those discussed in the labs, to connect to a database. The only difference this time is that the path of the database is provided via the command line argument instead of a prompt.

The **interface()** function is the login screen that provides options for both customers and editors to login. Both classes of users will be able to login using a valid id (respectively denoted as cid and eid for customers and editors) and a password, denoted with pwd. Unregistered customers will be able to sign up by providing a unique cid and additionally a name, and a password. Users will be able to logout, which directs them to the first screen of the system. There is also an option to exit the program directly. All calls to menus for customers and editors are made in this function.

The **customer(cid)** function allows a customer to choose what they wish to do within the program. They can choose to type '1', '2', '3', or '4' to respectively start a session, search for movies, end watching a movie, or end their session. Any other input allows the user to log-out and return to the log-in screen. The customer will be allowed to continuously select options until they choose to log-out. Also, if a session has been left open when the user chooses to log-out, the function will check for this and ensure that it is closed before the log-out is complete.

startSession(cid) is called when a user wishes to start a session. Based on the user's cid, they are allowed to start a single session through this function, which will be created without any further input from the user required. Within this session, a unique id will be assigned to its sid, and the current time and date will be recorded in its sdate. The duration is set to *None* as it is currently unknown when the session will end. The user will be denied if they attempt to start an additional session while one is already active. This is based on the specifications of the project, which limits the program to one session active at a time.

searchMovies(cid) function is to allow customers to enter unique keywords separated by space. The system will display a list of movies' title, year, and runtime, where they have a match in either the title of the movies, the cast's names of the movies, or the casts' roles of the movies. The list is ordered by the number of keywords matched and at most 5 movies will be displayed at a time. Then the customer can choose numeric options. With 0, the program will display another 5 movies if there are more movies that matched the keywords. If there's none, the program will just display the same movies again and again. The customer can also choose indexes from 1 to n, with n = # of matched movies. Where each index represents a movie, if the user chooses invalid index, the system will prompt the user to choose again and again. By

choosing such a movie, customers can see more information about the specific movie which includes casts, and number of people who watched the specified movie. In this stage, customers can enter 0 to start watching the movies and the time will be recorded. A message is prompt if the customer is already watching the movie. Customers can also enter a valid integer that corresponds to a casts' index to start following a cast. A message is displayed if the customer is already following the cast. At the end of function, customer will be returned to the main menu which is the customer(cid) function.

endWatchingMovie(cid) function is to allow the user to end a movie in the current log in. Since we only allow users to watch one movie at a time, when a customer selects this option in customer(cid) function, the system will automatically end a movie. The duration of the movie will be set to 'now' minus the 'startTime' in minutes. If the user is not watching any movie, a message will be displayed and it will return to the main menu which is the customer(cid) function.

endSession(cid) is called when a user wishes to end their session. The function will first check that there is a session to end, and if there isn't, the attempt is denied. If there is a session to end, then the function will calculate the time passed between starting the session and the moment in which endSession was called. This will be stored in the session's duration in minutes. The function will also check whether or not the session currently has a movie active within it. If not, then the function will end there. If there is a movie that has not ended, then the function will also end that movie, also calculating its duration in minutes based on the difference between the startTime of the movie and the current time. This difference will not exceed the overall runtime of the movie watched. Afterwards, now that the session is closed, along with the movie if necessary, the function ends without any further action.

editor(eid, pwd) serves as the menu for editors where editors can choose to add a movie or update recommendation. It is where the editors will end up after exiting the addMovie() or updateRecommendations().

addMovie() is a function in which the editor will be able to add a movie by providing a unique movie id, a title, a year, a runtime and a list of cast members and their roles. To add a cast member, the editor needs to enter the id of the cast member, and the program will look up the member and will display the name and the birth year. The editor can confirm and provide the cast member role or reject the cast member. If the cast member does not exist, the editor will be able to add the member by providing a unique id, a name and a birth year. We assume that the user knows the type of data or option they are suppose to input, but the function itself does provide error checking such as domain check, invalid option check, etc.

updateRecommendations() is a function in which the editor will be able to select a monthly, an annual or an all-time report and see a listing of movie pairs m1, m2 such that some of the customers who have watched m1, have also watched m2 within the chosen period. Any such pair will be listed with the number of customers who have watched them within the chosen period, ordered from the largest to the smallest number, and with an indicator if the pair is in the

recommended list and the score. A subroutine updates() is called for following operations. This is also where the editors will end up after exiting updates().

updates() is where the editor will be able to select a pair and (1) add it to the recommended list (if not there already) or update its score, or (2) delete a pair from the recommended list.

(c) your testing strategy

Testing included attempts at various combinations of options in varying orders within customer/editor's tasks and ensuring that intended responses were given by the program.
We test first by inputting regular inputs that follow specifications of the program.
Since we are also required to take into account when a user accidentally give incorrect input, we account for the following cases:

- When a user enters invalid inputs (such as inputs that will destroy the functionality of the program, e.g., data type error, options other than those provided by the program).
- When a user tries to watch a movie that is being watched in the same session.
- When a user tries to follow a cast that they are already following.
- When an editor tries to enter in a score value outside of the [0.0, 1.0] range

Most bugs we encountered are either during query testing or error checking. Since we need to account for a lot of error checking, we need many breaks and most bugs come from incorrect placements of breaks.
The program is tested on the lab machine and it runs successfully.

(d) your group work break-down strategy.

Weiguo Jiang: connect(), interface(), Tasks # 1 and # 2 of editors, README.txt. 10 hrs

Liam Chen: Tasks #2 and #4 of customer specifications. 10hrs

Dylan Clarke: Tasks #1 and #4 for customers. customer(cid) interface. 10hrs

All of us participated equally in creating the Report.pdf.

We communicated online to give each other updates about how our tasks were progressing. We coordinated to make sure that each part was completed early enough so that we would have ample time to merge them together, test, make adjustments, and prepare its final version.