DESIGN DOCUMENT FOR MINI-PROJECT 1

Introduction to Project

- In order to run this program, you must run it in terminal with 'python3 main.py <database>
- Once connected to a valid database, the user will be prompted to login, register or quit
- New customers can create a customer account and will be prompted to enter a valid username and password. Once completed they may login to access the customer menu
- Existing customers and editors can login using their username and passcode, which if correctly entered will take them to their respective menus

User Guide:

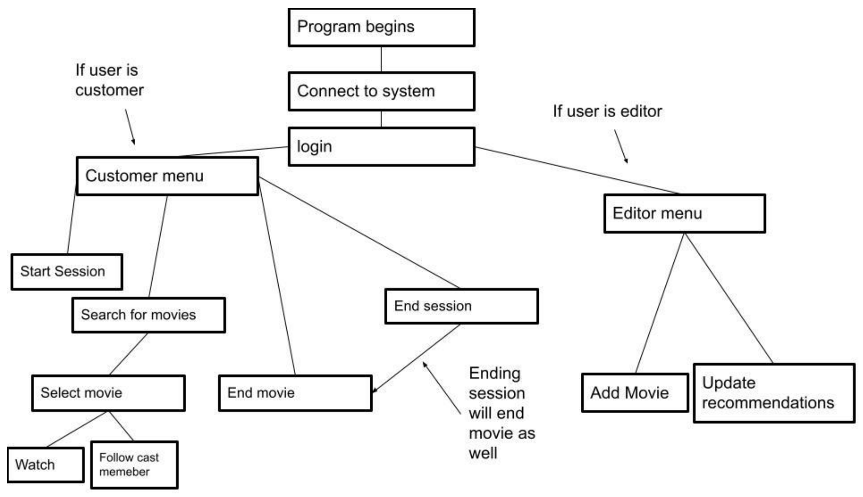
- Once a user successfully logs in, they can pick of one of the following options

1. Start a session: Begins a session for the user. Only one session at a time can be ran
2. Search for movie: Automatically begins a new session for the user. The user can search for movies using inputted keywords and they recieve details about the movies with the most matches(max 5 at a time). The user can request for more entries that match the keywords by inputting '>' and it will continue to reveal 5 at a time until there are 5 or less entries left, which then will print the remaining entries. The user can also then enter the movie id to get a list of cast members they can follow or begin to watch the selected movie or go back to the main customer menu.
3. End watching a movie: The user can stop watching the movie. And will be prompted back to the customer menu. The user can watch another movie in the same session that hasn't been watched in that session, but it is advised to only watch one movie per session.
4. End session: Ends user session

Editor Guide

- Once an editor successfully logs in, they can pick of one of the following options

1. Add movie: editor will be prompted to enter details about the movie, include mid name runtime etc. The user can also add cast members that already exist in the database. If that cast member doesn't have a pid in the database, the editor can add it that cast member into the database and fill in the cast member details
2. Update recommendations: The editor can select an annual,monthly or alltime report of all movie pairs watched within each time frame by distinct customers. It will list all pairs that fit within the time frame and display movie 1, movie 2, #customers who watched, and the recommended score for movie 2 if movie 1 has been watched. From here the editor can select a pair from the results(if reports aren't empty) and either add them into recommended, delete them from recommended, or update the pairs score.



Here is a general flow of how the data flows through the program

Description of Each of The Primary Functions
*for additional information on top of this guide, refer to the docstrings and comments within the source code.

- connect(path): connects the program to sqlite3 using the path variable which is the command line database entered. Initializes the connections and a cursor as well
- login(): based on what the user input, either log in, register or quit. If the user chooses to log in then the supplied user name and password will be checked for in both the customer and editor tables. If either of those queries returns a row then we log the user into either the editor or customer menu, wherever the details were found. If the username/password combo is not in those tables, then a message is displayed stating that. If the user registers then they will be prompted to enter a unique login and if the login has not been taken as well as in the correct format then they are prompted for a password. After registering the user can log in as a customer. If the user chooses to quit then the program closes
- customer_menu(): called from login() when the login details are in the customers table. This function acts as a hub for all of the customer functions, calling specific functions based on if the user starts or end a sessoin/movie or searches for a movie. sessions/movies cannot be ended if they haven't started and searching for a movie automatically starts a session
- editor_menu(): called form login() when login details were in the editor table of the database. Just like the customer menu this is the hub for all editor functions, editors can select to add a movie or update recommendations
- add_movie(): Called from editor menu. Adds movie into database. Prompts editor to type in a movie id(mid). Mid must not be in the database already. Then the editor will fill in the required information for movies such as runtime, name etc. After a movie is added, cast members can be added too. Editor supplies a movieperson id (pid) and if pid is in the moviesPeople table then it will print out the actors name and if the editor wants to add them or not to casts. If yes then the cast table is updated. If pid is not in moviespeople then they are prompted to add that pid into moviesPeople in which they will out the rest of the details (name, birth year) and update casts.
- update_recommendations(): Called from editor menu. Users select either a monthly, yearly or alltime report of movie pairs watched. Depending on which one they pick the selected timeframe will get queried and return the result pair and number of times watched. If the query returns the result then we check each pair to see if it is in the recommended table and return either 'x' or the score into a list with all the scores with each index representing each pair returned from the query. Then this information is all printed out. The user can then select one of the pairs (they are numbered from 1 to x) to either delete from recc, update or add. If they delete, then the program checks if the pair is in recommendations. It deletes if it is and returns an error message if it's not in. The same goes for add but instead it adds if it is not in recommendations with a score between 0 and 1. If the update pair is in recommendations then the editor can update the score between 0 and 1. For update and add if the scores are not between 0 and 1 the pair is not updated or inserted in the table.

The following functions are all called from user menu

- start_sessions(cid): begins a new session for the customer and inserts it into the sessions table

- search_for_movies(user_id, sid): sid=sessionid, user_id = user id. Allows a user to query for movies based on a list of given keywords that are either in movie title, cast role or movie persons name. Returns 5 movies at a time ordered by number of matched to the keyword per movie. User can choose show show more until user either quits or selects a movie id. Once they select a movie select_movie(selection, user_id , sid) where selection is the mid of the selected movie and the other 2 parameters are are the same as search_for_movies
- select_movie(selection, user_id, sid): Parameters were explained in search_for_movies(). Queries for all the cast members and the number of times the movie has been watched and displays them to the user. From there user can follow a cast member that calls function follow_cast_member(cast_names, user_id) or watch a movie which calls start_watching(sid, user_id, selection).
- start_watching(sid, user_id, selection): returns a tuple that contains information needed in the end_movie and update_movie functions. Program queries for the movies duration and the current time which along with the parameters of the function go into the return tuple
- follow_cast_member(cast_names, user_id): called from select_movie(). cast_names a list of cast_members of the movie selected in select_movie().Queries to double check that cast member is a valid choice and then inserts the user and cast member pair into follows
- update_watching(watching): Watching was the tuple of info from start_watching(). Checks to see if the movie has been and calls end movie it has
- end_movie(watching, endTime, userActivated = True): End the movie being watched and the duration of the movie watched and if it was longer than the runtime of the movie. If it was then duration = runtime else duration is unchanged. Update the watch table with duration of movie watched
- endSession(sid, cid, watching): ends the current movie and update the session duration

Testing Strategy

- Start with a barebones database (enough to log in from both user and editor) and check all the options to make sure there were no errors occurring in the python program when there was next to no data.
- Then slowly we tested some basic cases to see if all our options in our program were operating with a decent set of data.
- Added some test cases. Examples/results are
    - Movie pairs watched by customers 366 and 31 days before current time and see if they end up in our yearly/monthly report
        - They were found in the alltime report but not the others
    - Adding a movie id that was already in movies
        - Prompts user to try again
    - Ending session and ending movie without starting a session (tells you to start ses)
    - Checking case sensitive/insensitive searches when searching for movies (both work)
    - Logging in with invalid details or registering with a taken login(prompts user to try again)
    - Finding movie pair that hasn’t been added to recommendations and adding, updating and deleting them at the same time and checking the recommendations table after to make sure it is not there (table is unchanged from before)
    - Checking multiple keywords for search (works well)

Problems we ran into:

- Lots of cases where fetchall method on cursor returned an empty list, meaning you could not index it, needed to account for this
- Needed to make sure session/movie cannot be ended if they haven't been started
- Finding the correct query for finding the movie pairs watched within a certain time

Group Work Split:
Manav: design document , update recommendations, command line parsing, primary tester (~9.5hrs)
Josh: Start a session, end a session, end a movie, start watching a movie(split with Aidan),tester(~9-10 hrs)
Aidan start watching a movie, login and general menus, add movie, search movie, follow cast member,tester (~12-13 hrs)

To keep everyone on track, we set a firm time limit where all of our code had to be due, if not met then there it would be explicitly stated in this doc that the group member was not pulling his weight and was harmful towards the completion of the project