Mini Project 1 Design Document

A. Overview and User Guide

Our program starts with prompting the user to enter the path to a database. Then a login screen is presented, where the user has the option to log in, sign up as a customer, or exit the program. When a user signs up, they will be prompted to enter their name and credentials. When the user logs in, they will input their credentials, and our program will determine whether the user is a customer or an editor, and present the corresponding screen. The customer screen allows users 6 options: start a new session, search for a movie, end watching a movie, end a session, log out, and exit the program. Customers must start a session to search or end movies and can only have 1 active session at a time. Searching for a movie prompts the user to enter any number of keywords, and then exit the search by inputting "//EXIT//". 5 numbered results showing the movie title, year, and runtime are displayed and the user will have the option to view more results, see more information about a specific result, or return to the customer page. Viewing more information will prompt the user to input the number of the result they wish for. The number of customers who have watched that movie, and its cast members (numbered) is shown. The user can input "W " to start watching that movie or input the number of the cast member to follow them. Ending a movie on the customer page will prompt the user to enter a movie to end from a list of movies being watched. Ending a session will end all movies being watched and close the current session. If the user enters editor credentials on the login page, they will be presented with the editor page, where they will have 4 options: add a movie, update a recommendation, logout, and exit the program. Adding a movie will prompt the user to enter a unique movie id, title, year, and runtime. Then they will be prompted to add cast members. If the cast member already exists, the program will present their name and birth year. If not, the user will have to input the name, birth year, and role of the cast member. After all cast members are added to the movie, the user will be redirected to the editor page. Updating a recommendation will prompt the user to enter a time frame, monthly, annual, or all-time. This will show all the pairs of movies for the time frame selected. The user can then choose to delete, add, update the score of the pair, or cancel.

B. Design Components

Connect: establish a connection with a database from a given path

Login page: the user is prompted to either log in, sign up, or exit the program as a customer.

a. Signup: the user is prompted to enter a unique ID, name, and password. The customer table is then queried to check if the ID already exists. If so, the user is prompted to try again, otherwise, the program will insert the new customer into the customer table.
b. Login: the user is prompted to enter credentials. The program will query the customer table to check if any customers match the credentials. If there is a match, open the customer page, and save the cid as a class variable. If there is no match, query the editor table and check if they have any matches. If there is a match, open the editor page, otherwise print an error message and prompt the user to try again.

Customer page: the customer is prompted to start a session, search for a movie, end watching a movie, end the session, log out, or exit the program.

a. Start_session: the program checks that there is not an active session already, otherwise asks the user to end the current session first. When a new session is created, first the sessions table is queried to find the number of rows it has. A unique session id (sid) is created by adding 1 to the number of rows since the sid in the table is ordered and increments by 1. The current start time is set as a class variable (using the time() function from the time module). The sid, cid, current date (from the datetime module), and duration (set to NULL) is inserted into the sessions table.

b. Search_movies: the program checks if there is an active session, otherwise asks the user to start a session. The user is prompted to enter keywords one at a time (saved in a tuple), and input “//EXIT//” to finish the keywords. The keywords are then combined and used in a query on the movies, casts, and moviePeople tables to find all titles, years, runtimes, and mid of the movies which have any of the keywords in either the movie title, role, or moviePeople name. The results are ordered by number of keyword matches and a maximum of 5 movies are displayed at a time and the customer can see more if they want. The customer can see more information about a movie by entering its number on the list, which will call the movie_info function and pass the mid of the selected movie.
    i. Movie_info: based on the mid, a query is made on the movies and watch tables to find the number of customers who have watched that movie (watched is defined as the duration watched being over half the runtime of the movie). This number is printed. Then another query is made on movies, casts, and moviePeople tables to find the names and pids of the movie people in the movie, which are then numbered and printed. The customer can then choose to watch the selected movie, follow any one of the cast members, or return to the customer page. If the user chooses to watch the movie and is not already watching a movie, the mid and current time are saved, and a row with the current sid, cid, mid, duration (set to NULL) are inserted into the watch table. It will also add the mid to a list of movies being watched. The customer can also input the number of the movie person to follow that person, in which case the cid and pid are inserted into the follows table if it is not already there. The user can also choose to return back to the customer page.

c. End_movie: the program checks to see if there is an active session and at least 1 movie being watched, otherwise the corresponding error is printed. A query on the watch and movies table is made to find the movies being watched in the session. The customer can then choose which movie to end, which will calculate the duration watched (by subtracting movie start time from the current time, making sure it does not exceed the runtime) of the movie and update the duration of the movie in the watch table, and remove the mid from the list of movies being watched.

d. End_session: the program will check if there is an active session, otherwise prompt the user an error message. Ending session will end all movies being watched using the same steps as end_movie() and it will also calculate the session duration (by subtracting session start time from the current time) and update the sessions table with the duration for that session. It will also clear the current session id, and redirect customer back to customer page

Editor page: editor prompted to add a movie, update a recommendation, logout, or exit the program.

a. Add_movie: prompt the editor to input id, title, year, and runtime of the movie. If all inputs provided are valid, the id provided is used to do a query on the movies table to check if the movie being added already exists. If the movie does not exist, it is inserted into the movies table and add_cast() keeps being called with the id being passed in to allow for the addition of multiple casts until the user decides to exit
    i. Add_cast: prompt user for cast id and queries the moviePeople table to check if the id already exists. If it does, it prints their name and birth year. Otherwise, prompts the user to add the member’s name, and birth year, which are then added to the moviePeople table along with the id. Then it prompts the user to enter the member’s role in the movie, and adds the movie id, id, and role into the casts table.

b. recommended_page: promt the editor to choose a monthly, an annual, or an all-time report. After, a list of movie pairs will appear. The query is designed where a movie pair’s order is irrelevant to whether or not a recommendation belongs to it, and only unique movie ID pairs will appear. Only movie pairs which have been watched will appear. Afterwards, the user will have the option to delete a recommendation from a pair, add one to a pair, or update the score of an already existing recommendation. For all options, they will be prompted for a movie pair number. If they chose to update or add a pair, a score must also be inputted.

C. Testing Strategy

For testing we used the data prepared by Sriram Karthik A, Shreya P & Almer Muneer from Eclass with some of our own additions. Our general strategy consisted of first doing a simple test by looking at the database and manually figuring out either what the function/query should return or what the database should look like after the function/query is run for some given (valid) values. We then ran the program with the given values and checked if the program results matched our manual results. We then did more complex tests, such as trying invalid values (IDs that do not exist, entering letters when numbers are expected and vice versa, entering invalid characters), again doing a manual calculation before and comparing with results.Then we tested all the related functions together. Testing login page: first checked if valid eid/cid and password inputs go to corresponding page. Then tested invalid credentials. Then tested signing up by first creating a new user with a unique id and password. Then tested invalid ids (non-unique ids). Then we tested it together to make sure a new user can login and get directed properly. Checking customer page: first started a session, and verified that an unique sid, correct date, cid, was inserted into sessions. Searching movies was first tested just to make sure that correct movies were being returned by any number of keywords. Then to test the ranking, we put the keywords as the title of a movie,cast member, and/or role and made sure that specific movie was ranked number 1, followed by movies that shared words for title, role, and/or cast member, and ending with movies that only had a few matches, usually words like “the”. We also tested the case where there are no matches, which caused some bugs. For testing movie_info, we manually looked at the database after inputting a movie number and verified that the number of customers who watched the movie and the movie’s cast members were accurate. We also verified that when starting the movie, the values inputted properly in the watch table. We also tested starting a movie when the user already is watching a movie, and made sure the user could not start another one. When testing following cast members, we first tried to follow a person that the user did not follow already, again verifying by looking at the database. Then we tested by following a person the user already follows, and made sure the user can only start following a person if they do not follow them already. To test ending a session, we first started watching a movie, then waited a few minutes, and then ended that movie, verifying the values in the watch table. Then we tried ending a movie when the user had not started any movies, and made sure that was handled. We also tested starting a movie, ending it, starting another movie, and ending it. This revealed some bugs where previous movies that had already ended still showed up in the list of movies to end. This is fixed by slightly modifying the query to make sure it only selected movies that had a watch duration of NULL. For testing ending sessions, first we started a session and a movie, and then ended the session, made sure that the movies were being ended, and the session duration recorded correctly. Tested this again without starting a movie. Then tried ending a session when there was no active session. Made sure to handle all these cases. All inputs for functions are also tested with basic error checking (such as invalid characters, and list numbers out of range). To test adding movies, first we tested entering a movie id that does not exist. Then tested movie ids that do exist. Then tested adding letters when numerics are expected. Then tested adding cast members that exist and then tested cast members that do not exist. Then tested updating recommendations for each time frame. For each time frame we manually verified that the correct pairs are presented, and then tested adding, deleting, and updating their scores. All inputs were tested with invalid inputs as described previously. Most of the bugs we had were simple error checks, mainly entering list indexes out of range. These were solved by making sure the user could not enter those invalid indexes.

D. Group Work Break Down

First we worked together to create a starting template which included names of the primary functions we were going to need, and some helper functions like start() which just calls other primary functions . Faiyad decided to take the customer and login function, while Mark took the editor functions, sign up function, and connection function. Mark fully created the connect function (20 mins), and the sign up function (30 mins). Mark wrote the majority of the add movie functionality for the editors (1 hour), while Faiyad wrote some error checking for it and did some minor debugging (30 mins). Mark fully wrote the recommendations functionality (2 hours) but discussed parts of the finding pairs query with Faiyad (collaborative 25 mins). Faiyad fully wrote the login page functions (1 hour). Faiyad wrote the majority of the start session function (30 mins), which Mark wrote some error checks for and some minor debugging (took 10 mins). Faiyad started working on the search movie function but got stuck on the search query, and Mark came in to finish it up (Faiyad 35 mins, Mark 35 mins, 20 mins collaborative). Mark also set up the display results function for displaying 5 results at a time (25 mins). Faiyad then wrote most of the remaining portion of the search function, including the movie_info function and the start movie and follow functionionalty (1.5 hours). Mark added some error checking to movie_info and also did some minor debugging (15 mins). Faiyad fully wrote the end_movie function (45 mins) and end_sesssion function (took 20 mins). We both spent 60 mins together to combine all our work and make sure everything was working and also ensure consistency between our work. We did 1 hour of testing and debugging together and 30 minutes on our own doing testing and another 30 mins doing miscellaneous debugging. For the report, Faiyad wrote the test strategies (35 mins) while Mark wrote the overview and user guide, group work breakdown, and considerations (40 mins). We each wrote the details of our own functions for the design component (30 minutes for each of us). We each worked on our own sections on different files but everytime we both finished a significant component we combined our work, using the Live Share extension on VS Code. To stay on track we both set realistic deadlines like “finishing X sections by Y day”, and kept each other accountable.

Considerations:
If the user starts a movie, ends it, starts the same movie again, and then ends it again, it does not add the two durations together, and instead records the duration of the second time. Also the recommendations were based on the idea that the IDs in a movie pair are interchangeable and recommendations shown are not order dependent.