1. Please list out changes in your project directions if the final project is different from your original proposal (based on your stage 1 proposal submission).

The main difference is we haven’t implemented the share and compare functionality for a movie.

2.Discuss what you think your application achieved or failed to achieve regarding its usefulness.

Regarding usefulness, our application achieved the goal for users to search movies and TV shows by title. Users will get detailed information about the movie or show they searched. It also allows users to create a watchlist and add any shows they like to their favorite watchlist. We also allow users to write reviews for movies and shows. There is a completed user-login system so users can only do the things above when they are logged in. We also created some tables to show some interesting statistics about our database, such as the most popular movie or show for each rating.
Due to time shortage, we didn’t complete the share and compare functionality for movies or shows.

3.Discuss if you changed the schema or source of the data for your application

We don’t make any change to the source of data, but there is some difference in the schema, mainly in the watchlist table. Originally there are just a list id and a username attribute in the table, but later we find if users want to create a new list, they don’t have anything to input, considering list id is the primary key and automatically filled. Thus we add one more attribute, which is list name, for users to better identify their favorite list.

4.Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?

This question is similar to the last one. Regarding ER diagram, there isn’t any change to the relationship between 6 main entity sets, while we do change one attribute of watchlist table, namely the “list name” one. We do this because we want users to have a more clear identification of their lists. Besides, it’s better for users to create a list name instead of list id, since as primary key, there is higher probability to cause collision and fail to create the list by inputting list id. I believe the current design would be a more suitable one.

5.Discuss what functionalities you added or removed. Why?

We have almost added every functionality that we've planned. First, we added the general search functionality which allows users to input movie names and return the information. Second, We have create and login functions using username and password. Third, we have create, change and delete for favorite list creation, which allows users to create a unique list which stores their favorite movie. Users can also enter the list, add and delete the movie inside the list. Finally, users can add and delete reviews to certain movies. With these functionalities, the interface has everything the user wants in terms of searching movies, adding favorites and writing reviews. The only functionalities we removed is the share and compare function, which is mostly because of the urgent time schedule.

6.Explain how you think your advanced database programs complement your application.

The trigger prevents unqualified users from entering the database. Username and password can't include space as the first or the last character. The store procedure provides another easy way to display information on the front end with a simple call to the function.

7.Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project

1) One technical challenge we faced was the warning messages in the SQL trigger. We finally used "get" functions to fetch the message from the backend, and alerted it in the frontend. Specifically, when the SQL trigger throws out an exception and shows any warning messages, the insertion/deletion/update process will be terminated. That is to say, you cannot successfully insert/delete/update and throw a warning message from the backend at the same time.
2) One technical challenge we've faced was passing and displaying data using React. We were not familiar with the syntax, and were struggling to find where the data is passed as parameters and how the data is returned on the front-end. We tried different syntax and figured out that for some operation data is passed in parameters and for some it's passed in the body.
3) One technical challenge we faced was the implementation of the stored procedure. Since our stored procedure contains the cursor and two advanced queries, it is more complicated than the example we did in lecture. When using the cursor to loop through the database, we struggled on how to design the advanced queries and how to output the results. We overcame this

challenge by creating a new table for the result and insert result of each advanced query separately in the loop.

4) Another technical challenge would be how to prompt our users they have successfully created / updated their account / review / watchlist. Later we found out that under most circumstances, we can check the length of the response data, and for create function, if the length is 0, there might be something wrong. Similarly, we can also check whether the loginUser variable is null to require a user to first login in, then do any operation to their review or watchlist.

8.Are there other things that changed comparing the final application with the original proposal?

We don't have that much change comparing to the original proposal. Almost everything we have implemented is listed in our original proposal. The only difference is that we don't have the share or compare functionality and we have added one more attribute to the watch list.

9.Describe future work that you think, other than the interface, that the application can improve on.

We need to find several better csv files as our data set. Our current database is simply composed of different parts of one table. To be more specific, our Movie/TV Table, Director Table, and Cast Table are originated from one csv, so there is not much variation, and the meaningful attributes are limited. To improve our application, we can find one data set with movie info, one data set with director info and another cast data set with more complex information, such as which honor they are named, which type of movie/TV they are good at, etc. We can also find an independent company table containing information about what movies and actors/ actresses they have.
We can also add more functionalities to our application, by implementing more SQL queries and providing advanced searching options, like adding multiple filters and allowing users to search by multiple attributes.

10.Describe the final division of labor and how well you managed teamwork.

The whole project was composed of two major parts, which were designing the project and implementing the project. In the first part, everyone took part in the group work, including determining the goal of our project, designing a UML diagram, coming up with useful functionalities, and providing ideas of advanced queries, etc. The technical parts were divided into different portions to everyone, and all the group members were fairly distributed with coding works, like raw data processing, frontend and backend programming, SQL query/trigger/procedure coding, etc. Every week, all the teammates met via zoom several times, in order to keep track of the process of the project. It turned out that the project was accomplished chronically and decently.