Project Reflection: FlixFind+

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

   The direction of the project did not change much. There was a slight rework in our paging on the frontend due to adding in a login system, but other than that, it was just adding new features to the existing codebase.

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

   Our application successfully allows users to find movies by a specific platform (Hulu, Netflix, Disney+, Amazon Prime), which they can also filter by year released, age rating, average score etc. Users can also add specific movies to either a watch list or black list after logging in. We also implemented functionality to find the average user ratings for each movie, which would then output either a thumbs up or thumbs down icon.

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

   We haven't changed the schema or source of the data of our application. Our initial plan for the website was to allow users to search for a movie, filter by specific attributes, add to a blacklist/watchlist, and add user login as a creative component - all of which we accomplished in our final demo. Our initial source of data was also a Kaggle dataset which we continued to use throughout the project.

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?

   The ER diagram slightly changed to add a review table so average user ratings could be displayed for every movie. This table was created as part of the stored procedure that assigns review scores to each movie by calculating the average rating and assigning an enum for each movie. The original and final design stayed similar to each other due to us not changing the functionality from our original design that much.

5. Discuss what functionalities you added or removed. Why?
   We added the review functionality and we added encrypted passwords for our users. We added the review functionality to have an enum to display for each movie. We added

encrypted passwords to mimic what common applications do with passwords. We opted to remove some extra attribute functionality like times on the ratings and total watchtimes on the movie lists. This was due to lack of time and struggles to find the data to compensate for this.

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

   The advanced database programs within our application, both the trigger and the stored procedure, had unique functionality within our application. The triggers complemented our application in that we would create a new UserId and password for a user before inserting a new user into the database, and after the new user was inserted into the table we gave each user two unique list ids: one for the watchlist and one for the blacklist, and inserted these list ids into their respective tables within the database.

   The stored procedure for our application would calculate the number of ratings and average rating for each movie. Based on the number of ratings and the average rating each movie achieved, we provided a continuous movie review that was updated each hour. If the number of ratings provided for a certain movie was less than 3, the movie review was considered indeterminate. If we had 3 movie ratings or more with the average rating being greater than 2.5, we gave the movie a positive rating. Otherwise, the movie was given a negative rating. Since the stored procedure was run every hour, we were able to continuously update the ratings of every movie within our database, providing users with real time feedback on which movies they might find preferable.

   Hence, the triggers add needed functionality to our program to match a user with a userId, watchlistId, and blacklistId, while the stored procedure adds a cool feature/flair to our final product.

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.

   Raghav- We encountered a problem where the implementation of a movie selection query for features in the midterm demo was very different from the same query used in the final demo. This was due to us not accounting for how the functionality would change when adding users and lists. In hindsight, it would have been better to build the queries to adjust for the full functionality in the first place.

Madhu - When trying to implement multiple pages that linked together, we realized that we had only accounted for a single home page. To add another page, we tried to use the React Route, Switch, Redirect components, however, these were deprecated and gave us multiple errors when displaying the page. We then had to find the updated React components such as the BrowserRouter component to allow us to add a login page, following which we migrated our previous home page into a separate pages folder so we had a cleaner codebase.

Neha - We faced a challenge in passing the user information from the login page to the home page. To display the personalized user home page, we had to pass the user id, black list id, and watchlist id to the url. The home page then used the url to gather the userid, blacklist id, and watchlist id to get the user's rating for each movie and the movies on the user's blacklist and watchlist and display them on the UI.

Arnav - Before our midterm demo, we were unable to figure out how to provide a default rating for users if they hadn't provided a rating for a certain movie beforehand. We were looking to provide a default rating of 0 in this case, but we were joining the necessary tables using a natural join rather than a left join. Using the natural join, we were unable to see all the movies within the database, likely because the natural join was filtering these movies out due to the flawed implicit join clause. However, providing a left join fixed the issue as all records from the left joining table were returned.

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

   In the final application, we had more filters like years the movies were released in and a critically acclaimed filter that helps users find movies with higher ratings than the average on a platform. We also added a review column which shows if other users disliked or liked a movie based on the average user rating.

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

   Currently, a user can add a movie to both the watchlist and blacklist - in the future, we would like to update this to where a movie can only be added to one list, and if a movie is added to the blacklist we remove it from the main movies table. We would also like to add a friends option that allows users to find their friends and see what movies they have added to their watchlist and blacklist.

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

    Raghav and Arnav did all of the backend work while Madhu and Neha did all of the frontend work. The two teams were very intertwined together however, as both needed each other to debug connection bugs. Everyone worked on creating the queries and design for the database before we started implementing it.