Final Project Report

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

Our final project ended up maintaining many of the key functionalities we described in our initial proposal. This includes CRUD functionality as well as game recommendation functionality. The most significant difference between our proposal and the final project ended up being the actual user interface since our UI is more simplistic than the one we envisioned. We also decided to group user games together by an overall list and not subcategories of types of games they could be grouped together under.

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

According to our Project Proposal, the expected usefulness is defined as “can provide features like filtering and similarity scores to help them quickly pick up the games that they are likely to love.” Our website realized the expected functionality so we think our application achieved the usefulness.

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

We improved the table RecommendedGames to only maintain the necessary fields (GameID, GameName, SimilarityScore, UserID), and we altered the primary key to UID for identification purposes since we needed a unique identifier for every list of recommended games.

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?

We changed our relation between the Games and RecommendedGames table to be a one-to-many relationship because we derived a similarity score and a list of recommended games that have a high similarity score from one game in the user’s game list. The actual implementation of our RecommendedGames table also has a new primary key to identify the list of recommended games that are generated based off of one game. Our final design is more suitable to the scope of what we expected our application to accomplish since it recommends games based off of a game in the user’s game list rather than an arbitrary game. This means that it takes account of actual user data in order to generate these recommendations.

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

When looking at the functionalities in the proposal and the final product we changed a few things. One of the things we eliminated was the ability for users to be able to create groups of games that they want together. We eliminated this because games had a genre column which made it possible to group games on the backend. We also changed the recommendation system to recommend based on doing natural language processing on game descriptions rather than factors such as game title, description, star rating, developers. This allowed for a better recommendation system for users.

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

Our advanced database programs help complement our application by allowing for extra features for our users. For example, our game filtering system allows for advanced filtering which STEAM doesn't offer. This gives our application a unique characteristic which differentiates us from other game databases.

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.

Jai - Some technical challenges which we encountered was building a frontend for our application as none of us had lots of experience with frontend development. We learned how to use CSS and HTML to build a proper frontend. Additionally it was a challenge to connect that frontend to a Flask backend. We thought the video tutorial provided skipped a lot of steps in the application development process which we had to pick up on our own.

Satej - Another challenge we saw while implementing our database into our frontend was how to apply our trigger to be called on the addition of games through our frontend. Although our trigger was created in MySQL Workbench, we had to pass parameters into our web pages that confirmed that the game was already in the user's game list, and figuring out how to pass that specific game through in our Flask backend was difficult.

Qi - Other challenges such as connection issue and concurrency issue also needs to be handled carefully. For the former one, when deploying the application, it is important not to compromise the public key and only whitelist IP addresses that are completely secured. For the latter one, we do notice that when two users access the same data, dirty read might occur, which would cause problem when traffic is increased, which needs to apply proper transaction control.

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

Most of what we had planned out in our proposal remained true in our final project. The main difference is that users aren't able to create groups of games that they want together. Instead this functionality is encompassed under an overall list of games that a user has added to their list.

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

I think the application could be more efficient, especially the recommendation system. Currently the NLP model only gets generated when the user clicks on the recommended games button. Due to this it takes around 15-20 seconds for games to appear. We should make it such that once a game is added or removed from the users game list the recommended games are generated on the backend so that the user doesn't have to wait to see the recommendations. We could also improve the user interface so that it looks more professional and colorful, maybe add some features where users can define their own color mode.

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

We managed teamwork by setting up weekly meetings as well as clearly communicating using our team group chat. We did pair programming in Grainger where we would airplay our code editor to a TV and all work together to develop a feature. We would split parts of each feature up so that we could work more efficiently as well. This helped us move quickly while developing features.