Stage 6: Project Report and Demo Video

Project by: Jeffrey Yan, Milind Tarun Philar, Alexis Serrano, Peiqi Cai

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).

From our original project proposal, we did not add the feature indicating whether a user is a “parent” or a “child.” In our project proposal, we planned to label users to moderate content and filter searches, so underage users did not view mature content. Instead, in the final project, we decided to moderate content using the user’s age, not from a user label. Another feature we did not implement from our proposal was the user data visualization. Originally, our group decided to visualize collective data from all users. Our hope was to display what kinds of things the gamer community was playing, searching for, and popular genres/categories. In our final project, we decided to only display data from our stored procedure and advanced queries.

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

Our application so far has achieved the useful aspect of being able to ban games that shouldn’t appear when our users search through the list of games. Our application also has the backbone necessary to use data visualization through graphs and such because we have a working example of a graph of data using the Javascript chart library. One thing that we had failed to integrate that we had first set out with describing was the idea of having two types of accounts where one was a parent that could add games to the banned game list and one was a child which couldn’t access that list. We weren’t able to create permissions based on account type and so currently that would not be a secure functionality.

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

We didn’t change the source of our data for our application since from the beginning, we used the same dataset. We had changed the schema of our data immensely because we realized that it would be extremely inefficient to have one large table with every column on it. So we created tables like the Categories, Genres, General Game Descriptions, etc. in order to help us organize our data better and figure out where to store different pieces of information at any given moment.

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 followed through with the exact ER diagram that we had started with. The problem that we ran into when implementing our tables was that some of the values were off. We realized that if we stated that the values for a certain variable on our table were supposed to be booleans, the values wouldn't be loaded into the table properly. The solution we came up with for this problem was to instead store the true or false values as strings and just use string equivalence in order to determine whether a certain element was true or false.

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

We had initially planned on storing users' liked games and genres and having a wishlist for each user. The wishlist did not really fit with the other functions we implemented and would have been a separate feature, so we left that out. Instead of liked games, we ended up using "banned" games because it worked better with the search feature; we made an "exclude banned games" filter. Due to time constraints, we did not significantly customize the UI like we'd planned, instead prioritizing function. We also left out user types such as "Parent", also because of the limited time we had. Instead, the extra time was used to refine existing features and fix bugs. For example, a lot of time was spent fixing an issue where the user needed to double-click "Search" to display results, since it required learning about useState.

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

The trigger is very useful in updating two tables when a user is created, making sure that both LoginUser and Users contain the new user. The stored procedure takes our two advanced queries and puts the results in new tables, which can then be used to visualize the data in the form of a chart. This is one way to enhance our application by representing the data in a different way.

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.

Milind: Figuring out how to use HTTP methods when using JavaScript can be tough, especially when no one on the team has experience with JavaScript. I would probably recommend doing the project in Flask because Python is a simpler language and would most likely have a less steep learning curve when it comes to full stack development.

Peiqi: Processing data from a foreign source was a challenging task because there were different data types such as Boolean that were difficult to process. A simple solution was using the Varchar type to store “true/false”, which eliminated strange errors.

Jeffrey: One issue we encountered was useState, as seen in the CS411 ReactJS demo, updating asynchronously. This meant that sometimes, immediately calling a function that depends on an updated state would not result in the correct output. There are some ways to solve this, but the simplest was to just use variables, and pass the variable into the function to be called. The state could still be updated for use in another function later on, when it is guaranteed to have been updated.

Alexis: A technical challenge our group encountered was working with the Google Cloud Platform. It was challenging to create databases and optimize our database through GCP. The syntax and execution of the platform would be confusing and provide many issues when working in the GCP environment. Our advice for future teams is to work in the SQL workbench application. The environment is more manageable, easier to navigate, and easier to write queries.

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

In the original proposal, our group decided to create a user interface that displayed featured games based on user data. Our group had planned to use “user age,” games in the “banned games list,” and genres to display sections of games when the user logged in. Due to timing, we unfortunately could not display this content. However, we decided to display the games in a user’s “banned games” list. A major change from the original proposal is our overall user interface. We created a long thread of functionalities instead of individual pages for each functionality, such as login, search, and data visualization.

9. Describe future work that you think, other than the interface, that the application can improve on
   The following improvements can be made to the application:

- Integration of IDP services for login
- Adding more filtering conditions and price distribution filters for the game search feature
- Displaying a count and feature distribution of users' game bans
- Automatically filtering out sections that conflict with a user's age and preferences
- Extending the database and creating websites for users of different ages and preferences
- Adding new games to the database as they are launched
- Implementing an "add friends" method to let users add friends and support their chatting and collect subjective comments on users

10. Describe the final division of labor and how well you managed teamwork.
    The division of labor changes a little from our original proposal.

    Alexis: In charge of taking inputs and inserting them into user tables. Managed user data based on queries and in charge of updating and deleting from user tables based on user inputs.
    Jeffrey: In charge of creating the general search functionalities and filtering.
    Milind: In charge of front-end development and understanding the backend in order to allow for user interaction with our platform.
    Peiqi: In charge of creating data visualization and queries concerning data visualizations.

    Everyone worked together, Milind and Alexis also did the overall planning job; they were in charge of understanding the timeline of overall tasks and figuring out where they would need assistance, and then asking for help. Jeffrey and Peiqi also participated in some of the functional design of the database and debugging work.