Jack Gauer
Benjamin Blade
Sabir Rahman
Phillip Nguyen

Team007 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 main direction for the project remained the same, being a website where users can access different information about songs. Some changes went into the design of the website, which is touched upon later in this report.
2. Discuss what you think your application achieved or failed to achieve regarding its usefulness.
    - Our application achieved a song/artist database that allows the users to search for different songs, add them to their favorites, and use those favorites and a quiz to suggest new songs. It failed to achieve the deep statistics and graphs we originally intended to include, but the overall idea is still present.
3. Discuss if you change the schema or source of the data for your application
    - The source of the data was not changed.
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 the artist to song relationship from many to many, to one to many. We decided to do this because we wanted to treat collaborations as one artist.
5. Discuss what functionalities you added or removed. Why?
    - We switched out the entire ‘Home’ page just for a ‘Search’ page, and added an ‘Account’ page. Home page features were also not added. This was because we felt like the ‘home’ page was basically the same as the ‘search’ page, and the features that came with it were a lot more difficult to implement than we first estimated.
    - We removed the graph that tracks plays vs. time for each song. This was because it was hard to find data relevant for each of our songs to make a reliable graph.
    - We added the ability to favorite different albums, and after favoriting a song your favorite genre will be recalculated. The favorite genre is not put in by the user. We felt like you should also be able to add favorite albums and songs as it creates a more complete experience.
    - The ability to click on different songs to get more information has been removed as information is stated on the screen after a search. We did not want to have another button click just to the view the same information.
    - We added the ability to sort based on different elements after a search (Name, Artist, etc.) We wanted the user to be able to find songs they wanted easier, so adding this sort helps them find specific songs.

- The ability to see what movies/tv shows a song is in was removed. This was because we were not able to find reliable data that covered all of our songs.

6. Explain how you think your advanced database programs complement your application.
    - Having the advanced stored procedure for our quiz page will really help the user get songs recommended to them that are relevant. Having the advanced queries helps pull from a decent amount of numbers, and if they want their current favorites to recommend songs. This not only allows the user to browse different songs at their pace, it allows us to recommend new songs for them to listen to. Having triggers that automatically update 'favorite genre' helps the user know what they are favoriting, and also allows the website to recommend better songs to you.
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.
    - Jack: Originally when we had to merge both of our databases and populate our tables we struggled to do it efficiently. We were looping through our files way too much making it virtually impossible to populate the tables as it would take days. We were using functions like RAND() and other inefficient loops which will go through your entire database and are not efficient when it is a million lines long. In the end, we removed the RAND() and limited our data to a few thousand lines. This allowed us to actually populate our tables with data instead of waiting days. I would tell future teams to be careful about how you are looping through your data when populating, you do not want to be checking the entire database to just populate one row, it will not complete.
    - Sabir: One big issue was setting up the search bar for the website. When we originally had it set up, it was only working by having the user type in the artist's name correctly. Our original thought process was that we could have the query that runs in the html be surrounded by %'s to have it use like, but it was failing. However, it turned out that we needed to do this process on the javascript side. We ended up grabbing the search, saving it by setting a variable to '%artistName%', which was what the user typed into the search bar, then passed it into the sql query in the html, and that made it work.
    - Ben: When we added the functionality of the favorites count as an attribute, we needed to count how many favorites are inputted. The way we did this was with triggers that added 1 to it when a favorite was inserted and subtracted 1 when a favorite was deleted. The problem with this was that we had a trigger already to generate a favoriteID using the favorites count, and when you can remove favorites and decrease the count, you could get duplicate primary keys. I solved this problem by changing the favoriteID generation to a system that increments the previous largest favoriteID, such that the ID won't create duplicates because it isn't tied to a number that can be decreased.
    - Phil: When adding songs from the search and quiz pages into a user's favorites table we had difficulty keeping populating the favorites table for the correct user. A work-around we found was to set a global variable called globalUser on default

be set to guest. And whenever someone logged in it would change the globalUser to the userID. We could then access that globalUser and use it with a query to correct only show favorite songs for that userID. Additionally, the original query we used did not select the spotifyTrackIDs as they were not need that data to showcase the song information. However, the favorites table is only connected to the song table through spotifyTrackID and we need to spotifyTrackIDs to successfully insert information into the favorites table. So we had to adjust our query accordingly for the table we are trying to edit.

8. Are there other things that changed comparing the final application with the original proposal?
    - Everything that was changed was mentioned in the previous questions.
9. Describe future work that you think, other than the interface, that the application can improve on
    - The application could improve by having more statistics about each song to give a better suggestion to the user. If we store more information about each of the user's favorites or maybe songs they search up, advanced queries could use it to provide a more thorough collection of songs or possibly artists/albums.
10. Describe the final division of labor and how well you managed teamwork.
    - Jack: Back-end table creation/filling. Advanced stored procedure, quiz backend.
    - Sabir: Schema planning, Back-end dataset merging for table filling. Set up advanced triggers and functioning search bar
    - Ben: Front-end debugging, Trigger writing, Transaction
    - Phil: Front-end of the website, functional integration between the front-end and back-end, along with database retrieval or update capabilities, cheerleader
    - Everyone participated equally, with proper communication and teamwork. We helped each other on their parts to ensure that it was up to everyone's expectation of quality.

References:

https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_login_form_modal

ChatGPT Prompts used:

- How to connect data collected from an html page to a javascript file to store a global variable
- does this work with node javascript
- How do I open a modal animation window on startup
- how to make text appear in the modal when the user doesn't type the correct username How do I prevent page switching with app.post node javascript?
- How do I switch to a different page in a script in html?
- How do I make it so that when the script runs and the data is valid it will refresh the page?
- how do I call the getfavoritecount function when the page loads

- when i check my sql database it says the spotifyTrackID is NULL. I have a column called spotifyTrackIDin home page and my search function has Song.spotifyTrackID so how would i get function addToFavorties to get the spotifyTrackID?
- add buttons on home and favorites and quiz to add or remove favorite songs from those query results
- I have app.post('/remove-from-favorites', (req, res) => { const { spotifyTrackID } = req.body; const userID = globalUser; // Replace with your method to get the current userID if (!userID) { return res.status(401).json({ success: false, message: 'User not logged in.' }); } if (!spotifyTrackID) { return res.status(400).json({ success: false, message: 'spotifyTrackID is required.' }); } console.log('Removing favorite for user:', userID, 'TrackID:', spotifyTrackID); // Log details const query = 'DELETE FROM Favorites WHERE userID = ? AND spotifyTrackID = ?'; connection.query(query, [userID, spotifyTrackID], (err) => { if (err) { console.error('Error removing from favorites:', err); return res.status(500).json({ success: false, message: 'Database error.' }); } res.json({ success: true }); }); }); why am i getting an error