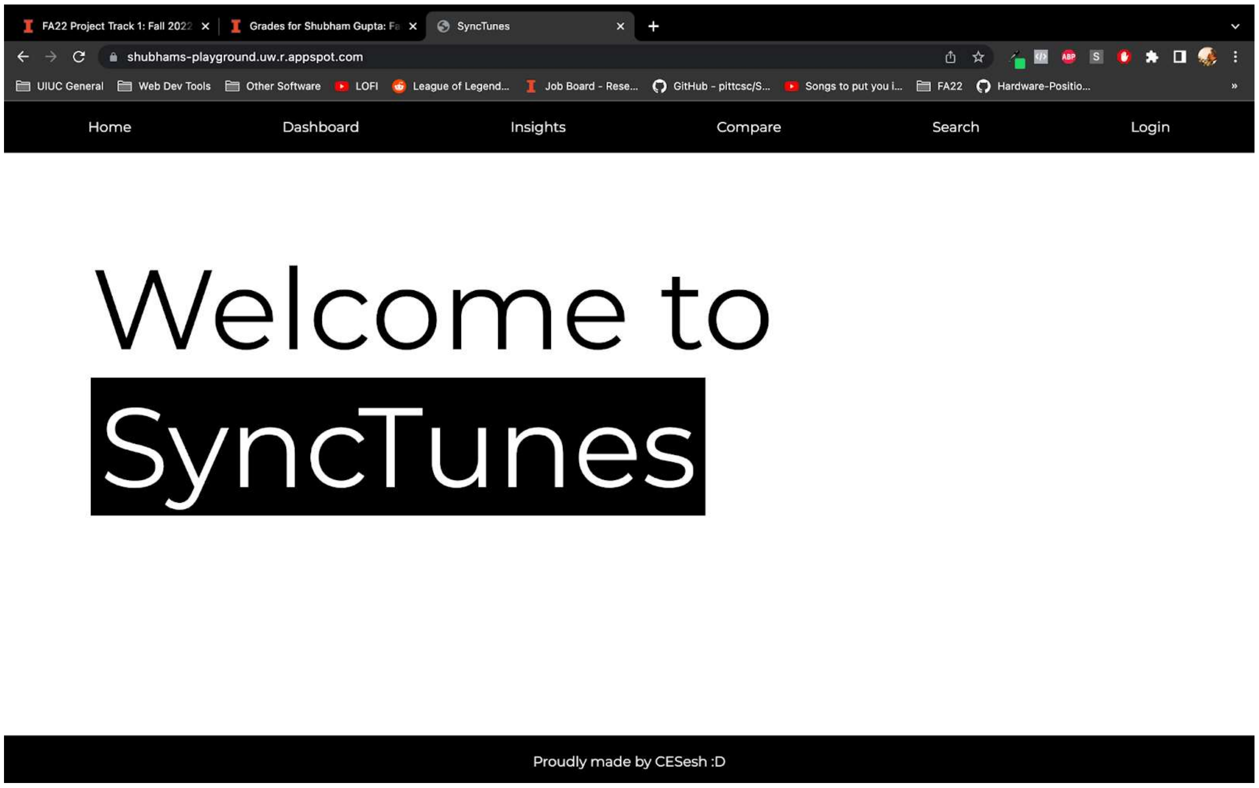
SyncTunes

Final Project Report

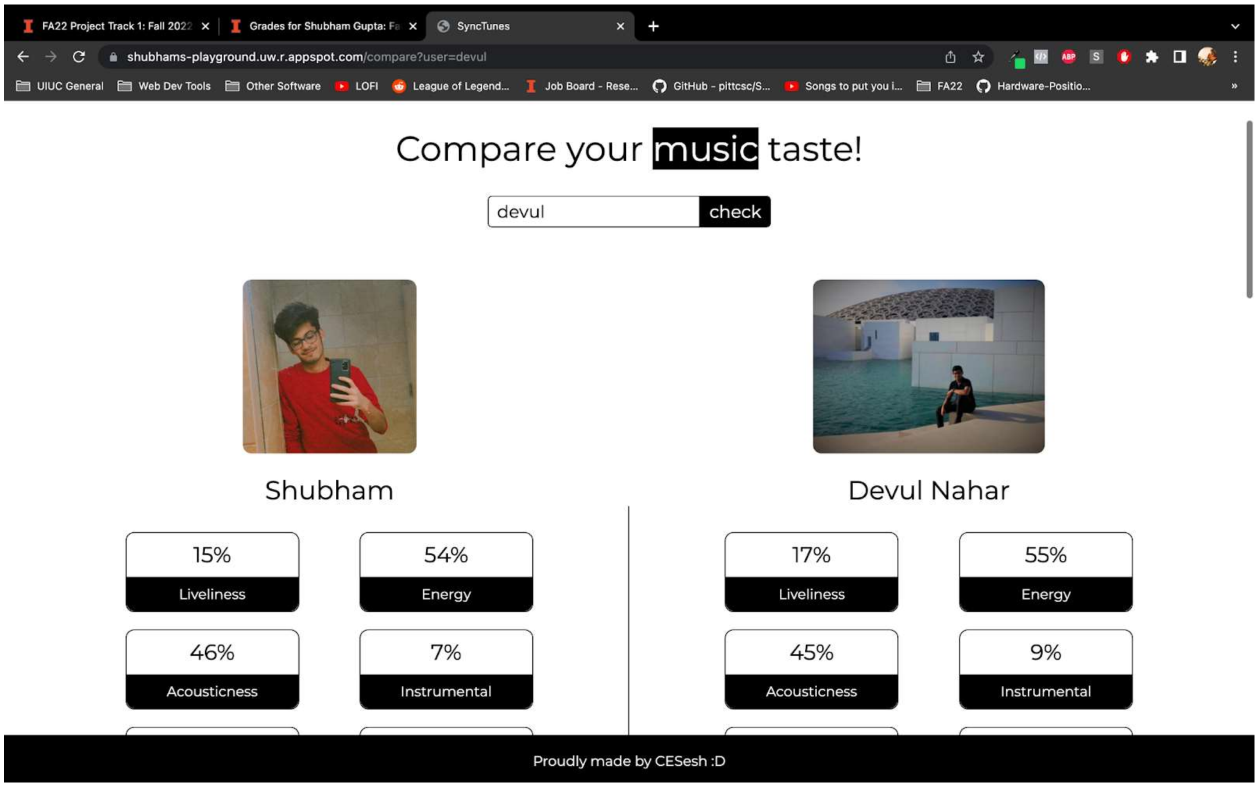
1. **Please list out changes in directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).**
   We achieved everything we wanted to achieve at the start of the project. The only major thing we changed was the interface of the design. In our proposal we were very unclear about how the website was going to look like. However, as we worked with new types of technology, we realized what was possible and thus greatly changed our approach. Here are some screenshots of our latest approach (you can see how much it has changed by looking at our initial proposal).

Youtube Video: https://youtu.be/1y1zMOSxSH4

Home Page:



Dashboard:



FA22 Project Track 1: Fall 2022 | Grades for Shubham Gupta: Fa | SyncTunes

shubhams-playground.uw.r.appspot.com/compare?user=devul

UIUC General | Web Dev Tools | Other Software | LOFI | League of Legend... | Job Board - Rese... | GitHub - pittcsc/S... | Songs to put you i... | FA22 | Hardware-Positio...

Compare your music taste!

devul | check

Shubham

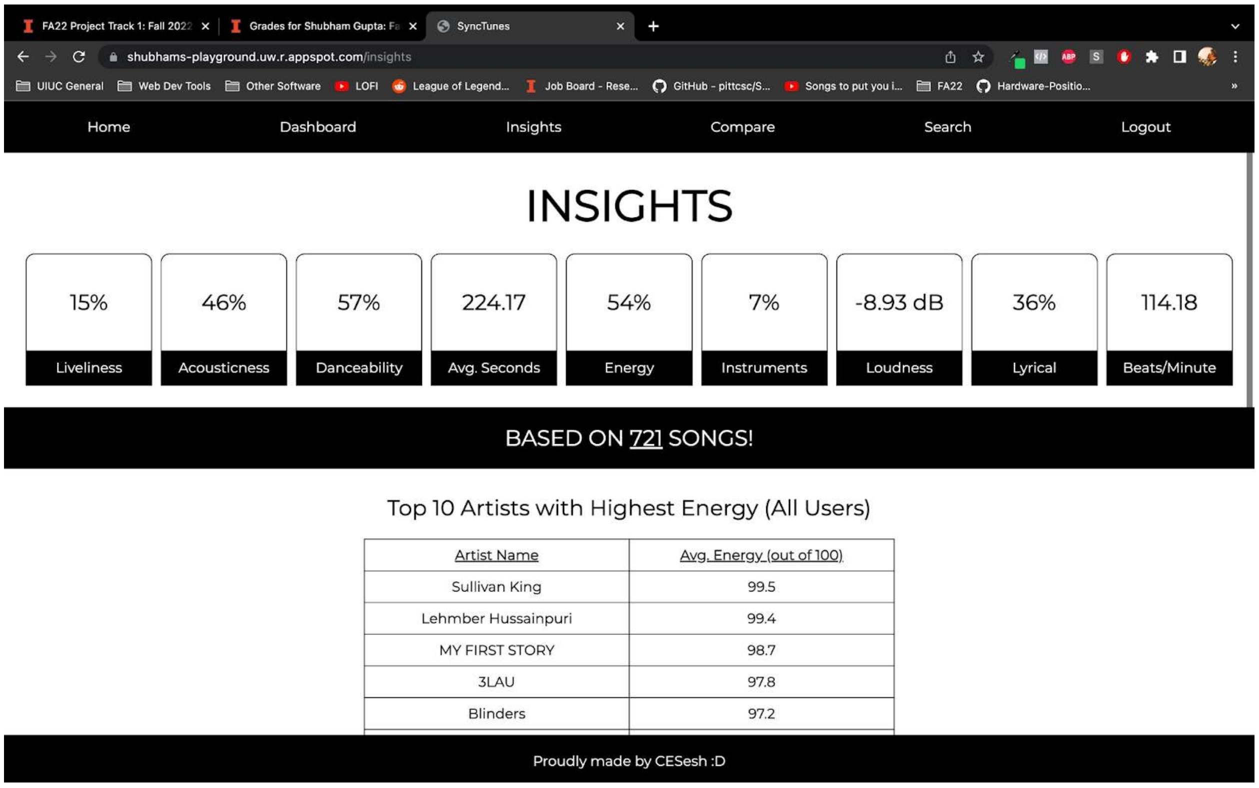
15% Liveliness	54% Energy
46% Acousticness	7% Instrumental

Devul Nahar

17% Liveliness	55% Energy
45% Acousticness	9% Instrumental

Proudly made by CESesh :D

Insights:



FA22 Project Track 1: Fall 2022 | Grades for Shubham Gupta: Fa | SyncTunes

shubhams-playground.uw.r.appspot.com/insights

UIUC General | Web Dev Tools | Other Software | LOFI | League of Legend... | Job Board - Rese... | GitHub - pittcsc/S... | Songs to put you i... | FA22 | Hardware-Positio...

Home | Dashboard | Insights | Compare | Search | Logout

INSIGHTS

15%	46%	57%	224.17	54%	7%	-8.93 dB	36%	114.18
Liveliness	Acousticness	Danceability	Avg. Seconds	Energy	Instruments	Loudness	Lyrical	Beats/Minute

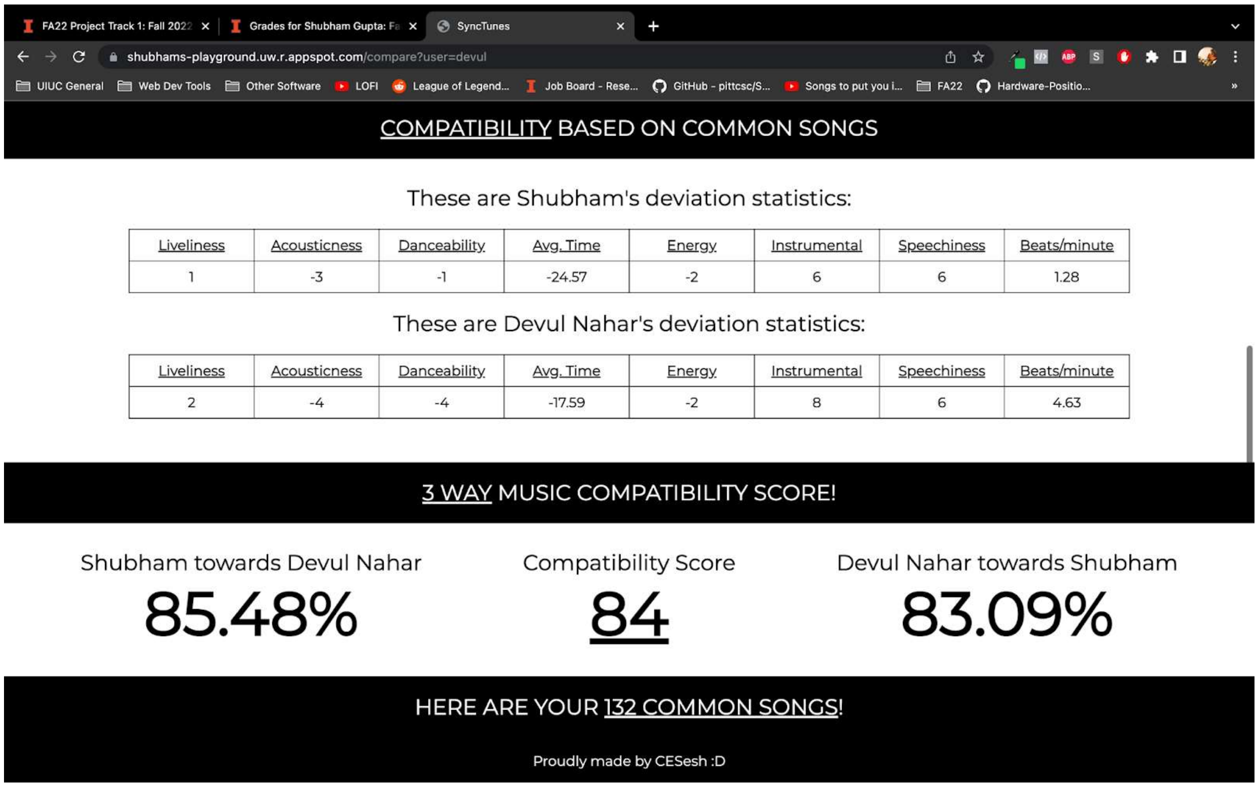
BASED ON 721 SONGS!

Top 10 Artists with Highest Energy (All Users)

Artist Name	Avg. Energy (out of 100)
Sullivan King	99.5
Lehmber Hussainpuri	99.4
MY FIRST STORY	98.7
3LAU	97.8
Blinders	97.2

Proudly made by CESesh :D

Compare:



FA22 Project Track 1: Fall 2022 | Grades for Shubham Gupta: Fa | SyncTunes

shubhams-playground.uw.r.appspot.com/compare?user=devul

UIUC General | Web Dev Tools | Other Software | LOFI | League of Legend... | Job Board - Rese... | GitHub - pittcsc/S... | Songs to put you i... | FA22 | Hardware-Positio...

COMPATIBILITY BASED ON COMMON SONGS

These are Shubham's deviation statistics:

Liveliness	Acousticness	Danceability	Avg. Time	Energy	Instrumental	Speechiness	Beats/minute
1	-3	-1	-24.57	-2	6	6	1.28

These are Devul Nahar's deviation statistics:

Liveliness	Acousticness	Danceability	Avg. Time	Energy	Instrumental	Speechiness	Beats/minute
2	-4	-4	-17.59	-2	8	6	4.63

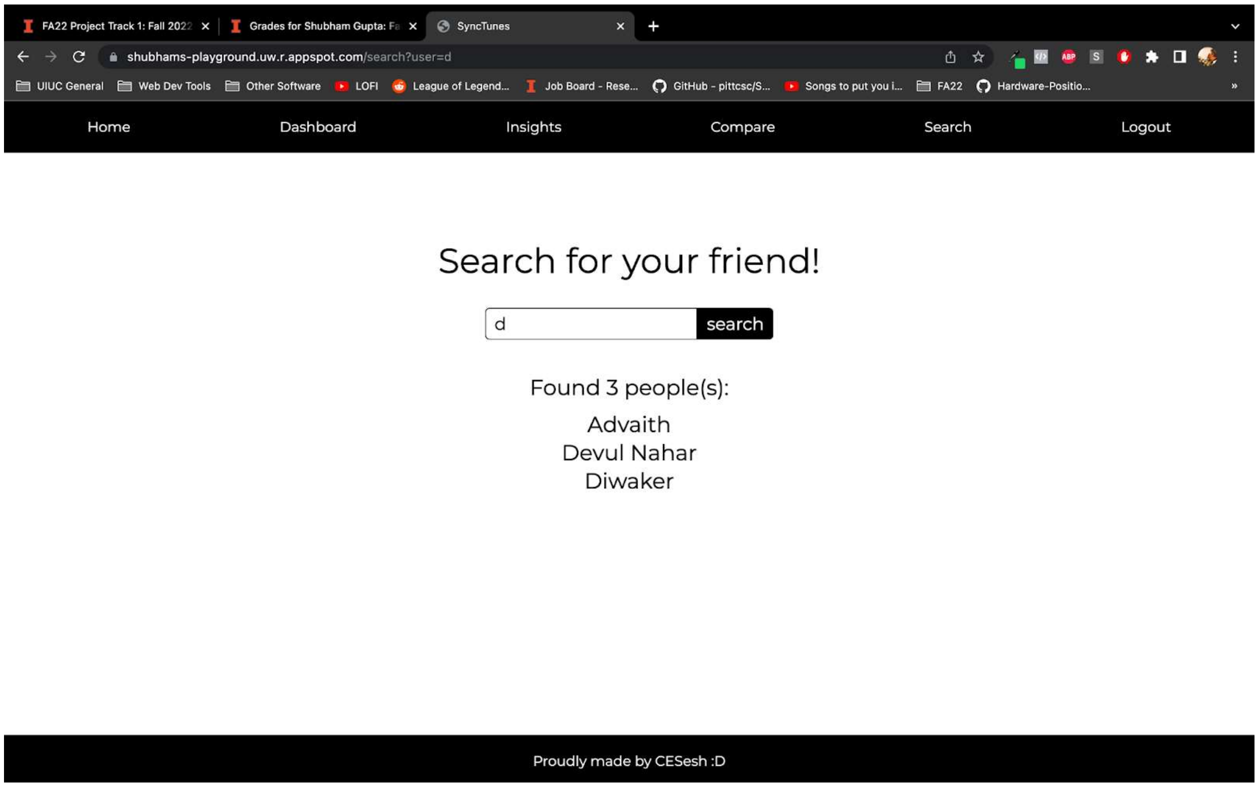
3 WAY MUSIC COMPATIBILITY SCORE!

Shubham towards Devul Nahar	Compatibility Score	Devul Nahar towards Shubham
85.48%	84	83.09%

HERE ARE YOUR 132 COMMON SONGS!

Proudly made by CESesh :D

Search:



FA22 Project Track 1: Fall 2022 | Grades for Shubham Gupta: Fa | SyncTunes

shubhams-playground.uw.r.appspot.com/search?user=d

UIUC General | Web Dev Tools | Other Software | LOFI | League of Legend... | Job Board - Rese... | GitHub - pittcsc/S... | Songs to put you i... | FA22 | Hardware-Positio...

Search for your friend!

d | search

Found 3 people(s):

Advaith
Devul Nahar
Diwaker

Proudly made by CESesh :D

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

   Our application was successful in strengthening the music community by helping people connect and make friends through common music taste. It was also successful in effectively using a cloud based solution to interface with Spotify's API to extract advanced audio features and do a detailed analysis of those features. We were also able to collect direct real time data from Spotify and analyze this data to give user insights.

   Though we completed everything we wanted to achieve in this project, one thing that we wish we could have done a bit better on is how we compare the scores between two people. As of right now, we simply use a weighted average of all the song features retrieved through spotify. We did not have a better algorithm to compare the music taste between people and would like to work on this problem if we have more time.

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

   We did not change the schema or the source of the data for our application.The schema and the source is constant since we are getting the data directly from the Spotify API.

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?**

   As we started creating and filling the tables in the database, we noticed that we were having some issues with the foreign keys and the on delete cascade constraint. We were unable to delete properly from the tables and started getting errors since we had defined multiple foreign keys in the table. We noticed that since we were using a couple of helper tables which would make the connections(joins) between the more important tables in the database such as Users and Tracks, the constraint was not working properly. In order to make sure that the helper tables (UserArtist, ArtistTrack) had distinct entries we decided to use a superkey and remove all the foreign keys, together with the on cascade delete constraints.This design was the most suitable design since it eliminated all our errors and made sure that the entries in the helper tables were unique.

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

   We did not remove any functionality that we had initially proposed. We did, however, add a couple of components to improve the user experience and at the same time fulfill the requirements of each stage. Additionally we added a creative component and the entire application is hosted in GCP. Here is a comprehensive list of all the things we added:

   - **"Fetch my data" button through AJAX** - The user logs in with Spotify. However, the only tables that get filled in during that time are the Users and the UsersAuthentication Table. Since the fetching of the entire data takes a long time(2-3 minutes) and since we want the user to agree in the fetching of their data first, we gave the user the ability to choose on whether they want their data fetched or not. Doing the fetch of the data with async javascript through AJAX gives an overall better user experience.

- **"Update your display name**" - This component fulfilled the requirement of having a query to update the database. It allows the user to change their username and see the change in the Dashboard page.
- **"Delete your profile" -** This is another component added which deletes user's data. When this button is clicked, we delete the data from the users table and then the delete operation on this table triggers a delete in all the tables where this user is found. This functionality makes sure that we neatly clean up the database in a delete.
- **Search -** Added a search box where you can type a string and it will return all the users with that username in our database.
- **Compare(creative component**) **-** While the basic requirements for this application were fulfilled with the insights page we also added a compare page. Each user has a unique code. This unique code is then shared with a friend and a score on how compatible your music taste is will be displayed, together with a breakdown of which are the artists and songs in common.

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

The database is how we store the data needed for the program. We used a mixture of advanced SQL queries, procedures, and triggers. This made designing and maintaining our database much more easier and efficient. For instance, since our application gave the users the option to delete themselves and their data from the database, a trigger was used to do exactly this. Furthermore, it is important to note that though many of the complex queries and procedures implemented in this application could have been implemented simply by using nodejs, it would not have been as efficient or easy to do. This is because SQL has been specifically optimized to do database related queries, so naturally it is a lot more efficient than using nodejs. Furthermore, since we had already learned how to do procedures and triggers in class, it was not that hard integrating them into this final project.

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.**
    a. Shubham - Using Log In with an OAuth session was challenging.The OAuth authentication scheme lets users submit credentials through an OAuth provider. The OAuth provider authenticates the user and sends an authentication response with claims about the user. Therefore signing in with Spotify and maintaining the session so that it memorizes that the user is logged in was challenging and it required a lot of research and reading of documentation.

b. Sara - I faced difficulties with the Spotify API. I had the issue of not getting the data correctly and sometimes the tables would get filled with null values. After doing some more research in the API I realized that it has a rate limit and you can only send a certain number of requests per second. Therefore, I could not just be spamming the API with requests. To solve this issue, I made sure that whenever I sent a request, I waited until I got a response and then continued with the next API request. My advice to other teams would be to try and research more about the limitations a technology has. That would be helpful in debugging.
c. Devul - Coming in I had no web programming experience. Therefore, I was mostly responsible for working on implementing the SQL queries and creating a basic frontend framework for the project. The frontend was kind of hard given that I had never worked with javscript/html/css before. It was quite a challenge learning complex topics and implementing them in the application at the same time. Specifically, it took a lot of time figuring out the correct syntax of language. This however got easier with time, and I got more comfortable working with these languages near the end of the project.
d. Diwaker - One of the difficulties I faced was modeling the database. Since we were scraping all the data from the Spotify API it was hard to model and decide which features we needed for our own project. Due to this reason and since we were using a large number of attributes, the table design was changed multiple times until we were able to get the schema that worked best for our application. My advice to future teams would be to think ahead and be adaptive to changes that need to be made in the project.

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

Despite the database schema changes, we mentioned above(section 3,4), we added functionality to support all the CRUD operations as explained in section 5 of this report. On the interface side, we added multiple components to complement our application such as animations and pictures of the common songs.

9. **Future Improvements**

Use Neo4J

Despite the interface, one of the improvements we would like to make in our application is using Neo4J, instead of SQL. After learning about Neo4J and all the advantages and features it has, we came to the realization that it fits best with our project. The data we retrieve from Spotify is highly connected. For instance a user listens to several artists, an artist has a certain number of tracks and each track has its own properties which we use in the insights and the compare part of our application. Therefore it would have been much easier to represent our data using Neo4J. As mentioned above, one of the main issues we faced is that we had to change the schema of our database several times until we figured out how to represent the data correctly. Switching to Neo4J would give us flexibility as we do not need to define a schema upfront, agility and performance as it is easier and faster to retrieve, traverse and navigate connected data. The queries were also highly complex and slow since we have multiple tables and therefore multiple joins.

More stored procedures

Before stage 5 of the project all the data manipulation and processing was done separately on the NodeJs side. Although we created an advanced procedure as it was in the requirements of Stage6, we realized that we can make use of more stored procedures. That would increase performance as stored procedures have the ability to reuse compiled and cached query plans.

Improve error handling and checks

We would like to improve our error handling by using try-catch blocks and use more Null checks on the data retrieved from Spotify. Additionally, we could give the user the option to see a load bar while the data is being fetched.

Using temporary table for compare

Currently, every single time a user wants to compare its data with a friend, we have to make all the calculations and then display it in the webpage. We could store all the data calculated in a temporary table to increase efficiency.

10. Division of labor and teamwork

For our backend, we used Node.js and SQL for querying to get the relevant data. Our team had a 50-50 split for frontend/backend development between two groups - (Sara and Devul) and (Diwaker and Shubham) respectively. Front-end and site development was handled by Devul and Sara, and was done through HTML and CSS. Sara and Devul were responsible for the design and programming of the web pages.

As for the more specific parts of the project, Shubham worked on OAuth login functionality, Diwaker worked on the REST calls to retrieve the data from Spotify, followed by all of the team work on querying this data. Sara and Devul also helped with API calls and the backend, once the frontend was done. We had weekly meetings where we updated the team on the progress made and worked together to make sure the requirements of each stage were fulfilled.