CS 411 Team 109 - Project Report

Sally Xue, Pradyumann Singhal, Lily Zhang, Joseph Schanne

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 initial idea was to have a place where we can store comments for the Spotify music. Our main idea is still the same. However, we did make some changes in implementation as discussed below in the report.

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

- Our website successfully achieves its minimum viable product (MVP) by enabling users to comment on and rate songs from Spotify, fostering an environment where individuals can express their opinions and engage in discussions.
- In comparison to similar platforms, our website offers additional features such as nested comments and the ability to rate songs, enhancing the overall user experience. Furthermore, the site is designed with a clean and intuitive user interface to ensure ease of use.
- At this time, the website does not include functionality for song recommendations based on user preferences. However, this feature is a potential enhancement we plan to consider in future development.

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

- In comparison to the initial proposal, we opted not to use the Apple Music API to retrieve editorial notes due to the high cost of a developer account. Instead, we supplemented the `Description` field in `ALBUMS` table with additional details about the tracks within an album, including their respective track numbers—data not originally included in the database structure.
- Furthermore, we updated the `Description` field's data type from `VARCHAR(255)` to `VARCHAR(5000)` to accommodate longer descriptions, ensuring sufficient storage capacity for more comprehensive information.

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 table structures remained unchanged, with the only modification being an update to the data type of the `Description` field in `ALBUMS` table, as previously mentioned.
- The `CreatedOn` column in the `Comments` table was updated from `DATE` to `TIMESTAMP` to enable validation that ensures no user adds a second comment within five minutes of their previous one.
- Similarly, we also added the `LastEditTime` column to the Comment Table to better manage the time restriction of editing comments.
- We opted to make the changes in the table because as we moved forward in the project, we realised we didn't consider how much precision we would need for some of the logic that we wanted to implement.
- While the initial design of the tables was pretty well-optimized, we believe the new structure is better suited for our project as it has more precision that allows us to include more complex logic to our software.

Discuss what functionalities you added or removed. Why?

Remove:

1. **Recommendation System**: The recommendation system functionality was removed as it was not part of the minimum viable product (MVP). As such, its exclusion does not compromise the core objectives of the project.
2. **Apple Music Integration**: Features reliant on the Apple Music API were removed due to the high cost of obtaining a developer account, making this functionality impractical within the scope of the project.

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

Stored procedures

- Comment Management with Time Restriction
    - A stored procedure has been implemented to allow users to add comments on songs while enforcing a time limit. This ensures that users cannot post multiple comments within a short timeframe to artificially inflate the song's rating.
- Password Update Functionality
    - A stored procedure is provided to enable users to securely update their passwords. It makes sure that the user knows the old password before switching to a new one for better security.

Transactions

- Unique User Identification
    - Transactions are used to generate a unique `UserID` using a random generator. This `UserID` serves as the primary key for identifying users during login and ensuring data integrity.
- Time Restriction when editing comment
    - A transaction has been implemented to enforce a 5-second time limit for editing a comment. If the 5-second check fails for any reason, the edit will not be applied, and the transaction will be rolled back, ensuring atomicity.

Constraints

- Password Length Requirement
    - A constraint enforces that passwords must be at least 5 characters long to ensure a minimum level of security.
- Foreign Keys
    - We are using foreign keys throughout the database to implement constraints on values with tables.

Trigger

- Dangerous Character Insertion Check
    - We have a trigger that checks whenever a password is created for the first time if it has any dangerous characters like single or double quotes within them.
- Dangerous Character Updation Check
    - We have a trigger that checks whenever a password is updated if it has any dangerous characters like single or double quotes within them.

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.

- **Pradyumann**: Writing Procedure while taking input variables and debugging. While we had gone over writing procedures in class, we didn't write any procedures that take input variables so figuring out the syntax properly took some time. Moreover, figuring out the need to use DELIMITER command to input procedure took a while.
- **Sally**: I was responsible for the initial setup of connecting the database, backend and frontend - it was my first time doing this so it was a bit challenging, especially how to connect our database hosted on GCP with python flask backend. I end up

using the `mysql.connector` library and the database credentials of our database, such as `host`, `port`, `user`, `password`, and `database` name. Then the connection parameters are defined in the `db_config` dictionary, and the `mysql.connector.connect(**db_config)` function initializes the connection. Additionally, the `host` IP could be found by: going to our database instance on GCP -> Connections -> Networking - Public IP connectivity - Public IP address, this also takes me some time to figure out. We also recognized that there might be better approaches, such as using a VM, but for the sake of time, the approach described above is the one I chose.

- **Lily**: Implementing Spotify authentication using OAuth2 presented initial challenges due to the lack of comprehensive online examples that included frontend code. This required me to determine which portions of the code should reside on the frontend versus the backend and how to establish the connection between them effectively. Additionally, as this was my first experience implementing page redirection with an external API, it was initially a confusing process to navigate.
- **Joseph**: This was my first time really doing web-development as a computer engineer, so I had some interesting growing pains to adjust to. Setting up the development environment was surprisingly high-friction for me as I've never written web code, let alone use tools like React or Flask, so I had to hunt down some system level dependencies like microsoft VC redistributable to compile MySQL, and had to change python interpreters, as flask/mysql require python 3.11, not python 3.12 (my standard interpreter). Beyond this, I had to dig into really writing javascript and using NPM for the first time, and had to learn how to use React to create frontend elements. A specific element of Javascript that caused me a slight headache was the routing table, as an element's placement in this structure is meaningfully impactful to the frontend's functionality. Thankfully my group members were very helpful, so I was able to learn a lot about fullstack development and practical usage of MySQL.

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

- Comparing the final application with the original proposal, no major changes were made. The core functionalities outlined in the proposal, such as commenting, rating songs, and Spotify authentication, were implemented as planned. Minor adjustments included removing features like the recommendation system and Apple Music integration due to practical constraints, but these were not part of the minimum viable product (MVP) and did not impact the overall objectives of the application.

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

- Implementing personalized recommendation features to enhance user engagement and discoverability of content.
- Providing additional search options, such as searching by albums or artists, to improve usability and navigation.
- Allowing users to leave comments on albums and artists, fostering greater interaction and discussion within the platform.

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

- Pradyumann: Implemented backend database CRUD logic and advanced database transactions, stored procedures, triggers and constraints.
- Sally: Initial setup of database (GCP), backend, and frontend connections; Implemented frontend styling; Developed the nested comments and rating feature; Developed the transaction feature for editing comments.
- Lily: Managed Spotify data collection, database setup and data import; Developed frontend pages and overall frontend structure; Implemented frontend and backend features related to Spotify authentication.
- Joseph: Developed registration and password reset frontend and corresponding login/logout/registration backend endpoints. Contributed to the creation of the unique UserID functionality.
- We effectively adhered to the project checkpoints, making consistent and reasonable progress at each stage. Additionally, tasks were distributed promptly following the final checkpoint, ensuring that each team member had sufficient time to complete their respective responsibilities.

The copy of all the advanced queries are in a separate file called “Advanced Database Feature Queries - Team 109”.