CS 3200 Project Final Report

Team Name: DuncanBRosasRWangM
Team Members: Brady Duncan, Robert Rosas, Michelle Wang

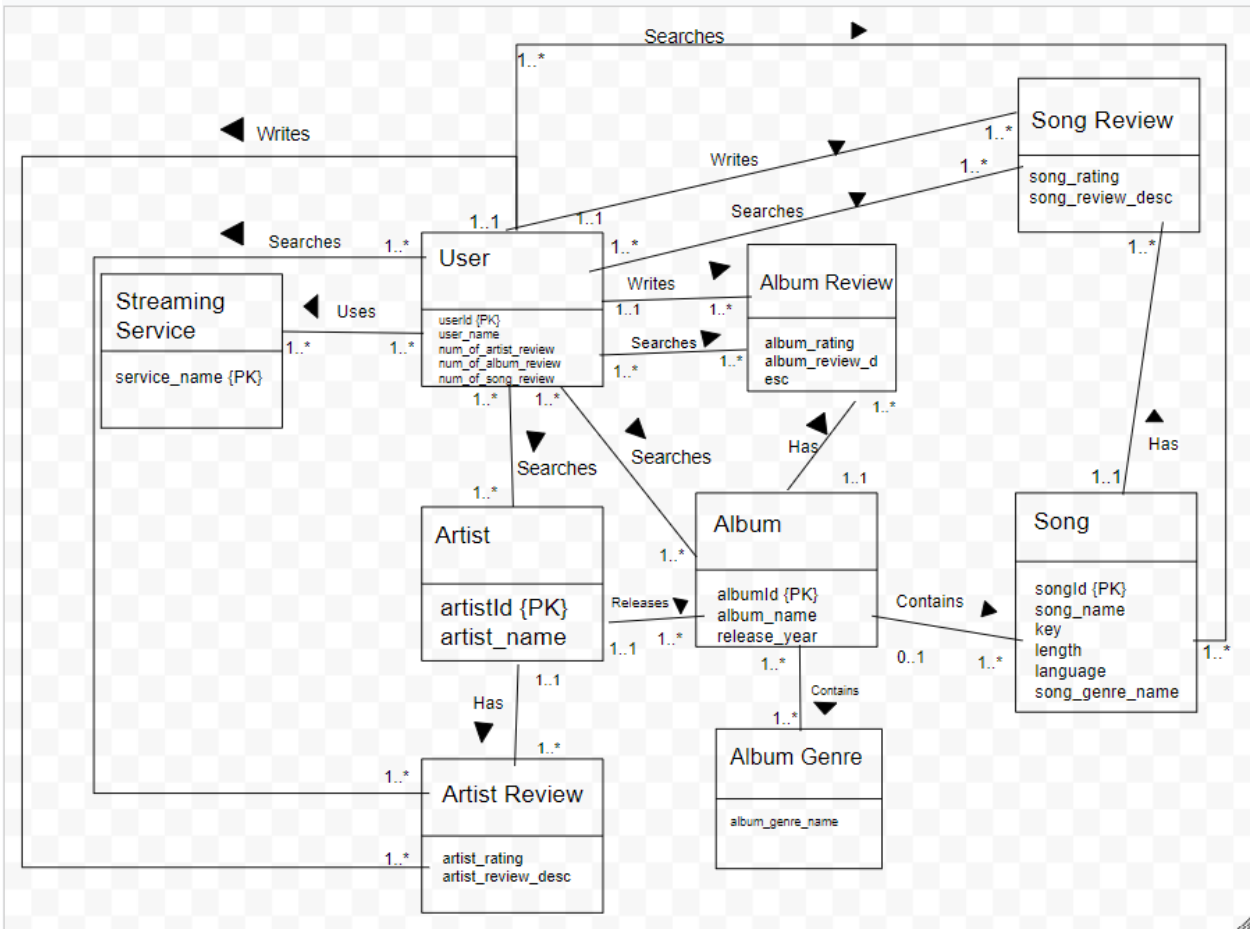
README

We used MySQL workbench to create our tables for our schema and procedures, functions, and triggers. The programming language we used was Python. The version of Python we have is Python 3.9. You can download it through this link: https://www.python.org/downloads/. We used PyCharm Community Version 2022.1.2. You can download it through this link: https://www.jetbrains.com/pycharm/download/#section=windows. Once we were inside Pycharm, we went to settings to download Pymysql and cryptography. Other installations that were already in our python interpreter included cffi, pip, pycparser, setuptools, and wheel. Please download the dump into mysql workbench in order to have our schema on your device. As for the application part, please download the three python files we have in our zip file: app.py, db.py, and index.py. Please make a configuration for all three of these files, but run the application under the index.py configuration.

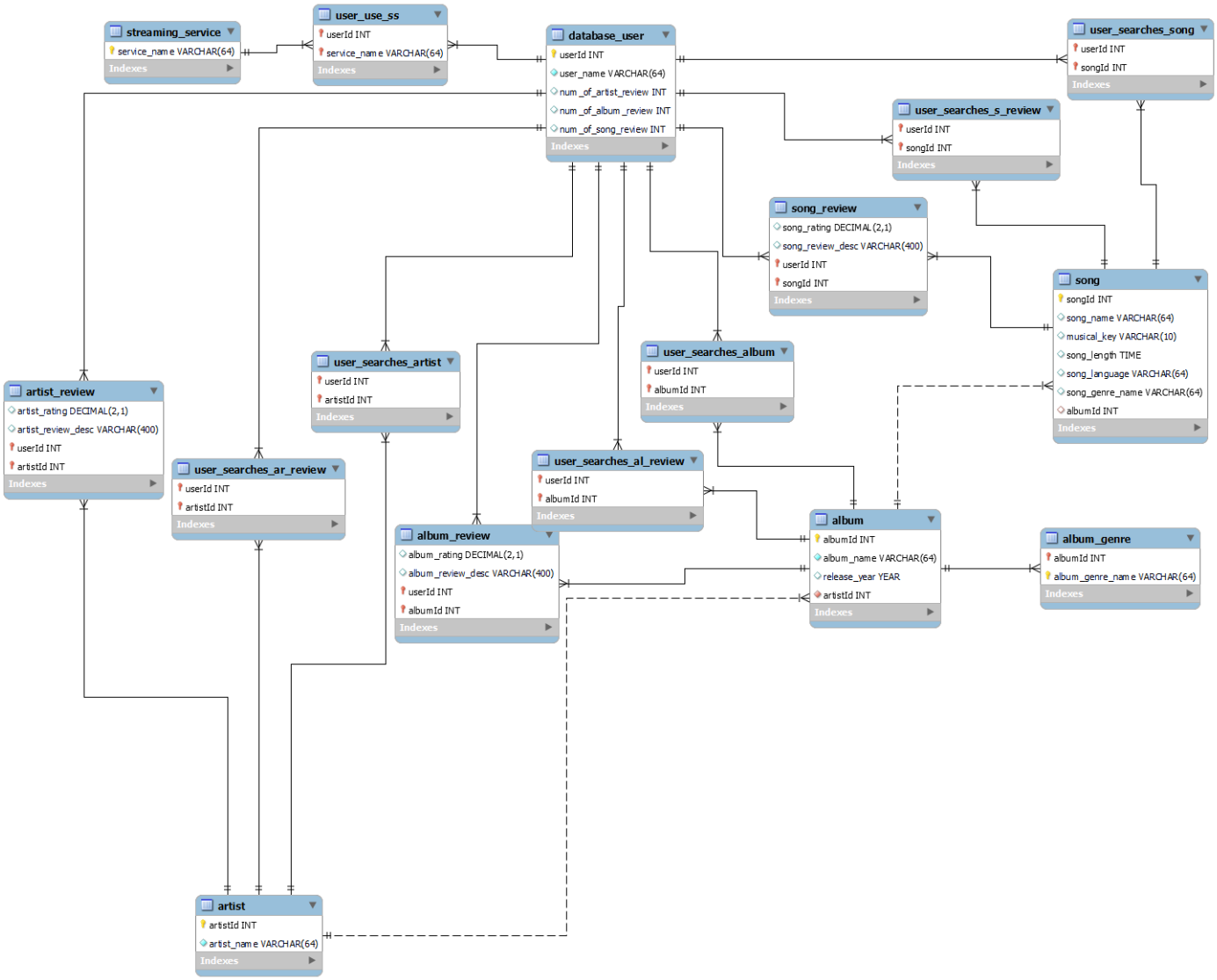
Technical Specifications

For this project, we created our database and procedures using SQL within the MySQL workbench. For the database application that allows the user to interact with our database, we used Python, using the PyCharm platform. We allow the user to interact with our database through the run window on the bottom of the screen in PyCharm, where we will provide good, easy to interpret feedback to the user based on their input. Our application allows the user to first make an account if they are a first time user. We also ask them to input the streaming service(s) they use. Then, we allow them with plenty of ways to interact with our database, whether that be through searching, adding, deleting, updating, and/or checking. Through searching, we allow users to search for artist, song, album as well as artist review, song review, and album review. Then of course, we also allow the user to add and delete songs and artist, song, and album reviews, but only be able to delete reviews that they made themselves. Users can also update the different reviews they made if they ever change their thoughts on a song, artist, or album. Lastly, we allow users to check statistics regarding themselves based on the number of song, artist, and album searches as well as song, artist, and album review searches. We also provide the user with information on the users per streaming service. The user can do as many actions on our application as they want before exiting out.

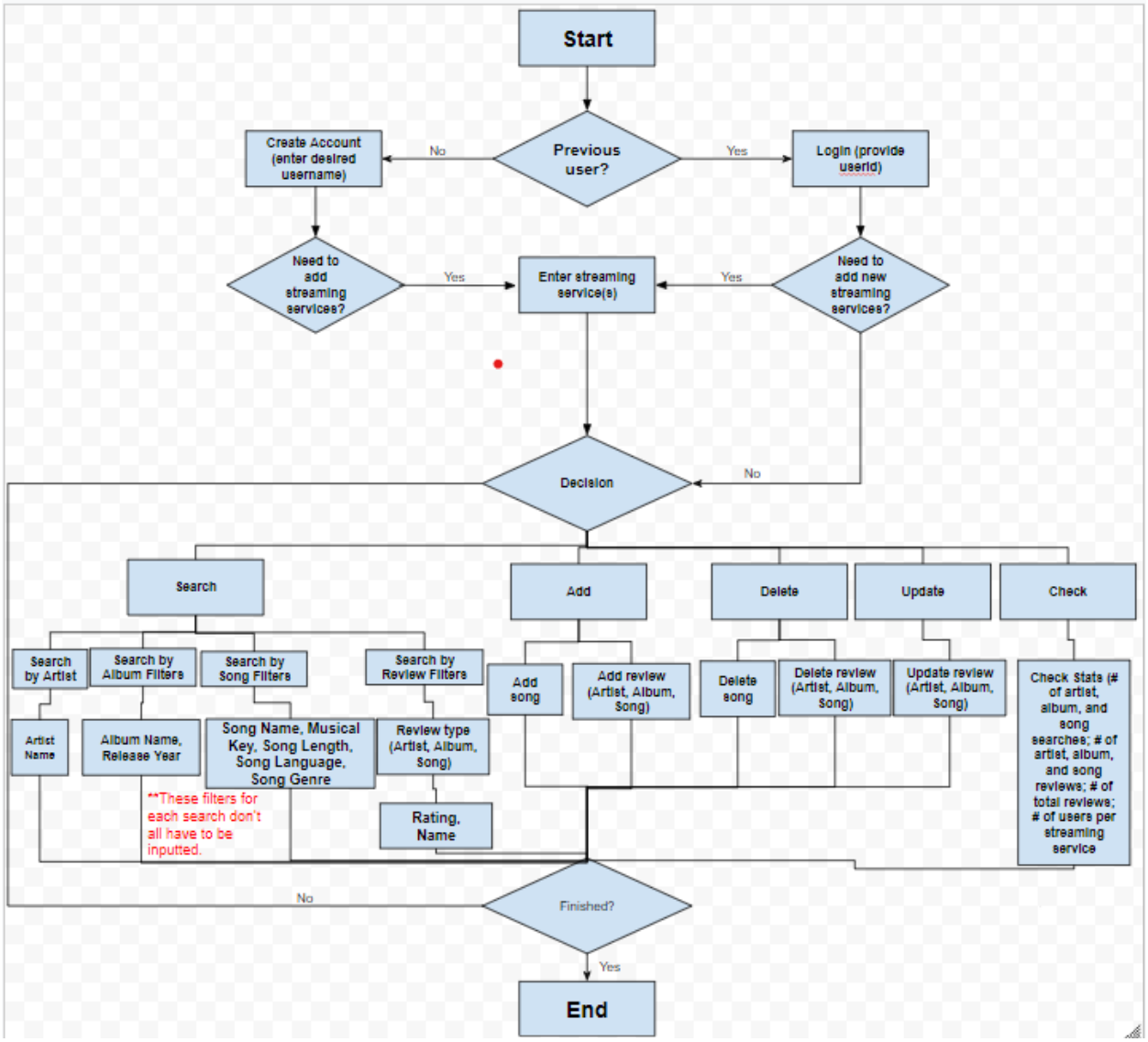
Conceptual Design - UML Diagram



Logical Design



User Interaction Flowchart



Lessons Learned

1. *Technical expertise gained*

Prior to this project, none of us were really experienced in either complex SQL applications or Python implementations of SQL databases. So, this project definitely pushed our boundaries and forced us to learn a lot through research throughout the entire project whenever we were met with something we didn't know how to do or were stuck on a bug for a long time. Especially with regards to creating procedures, functions, and triggers, we were not very confident on it at first. But definitely having to use it in our project allowed us to practice a lot, and by the end of the project, we felt very comfortable with coding procedures, functions, and triggers. With regards to our python skills, we came in with varying levels, but we all learned a lot and grew on our knowledge of python in general. Also, having not had any experience in coding both the database as well as the front-end portion to run together, this project pushed our limits and we were satisfied with the work that was achieved at the end.

2. *Insights, time management insights, data domain insights, etc.*

At first glance, the project appeared simple - all we needed to do was implement search and review functions for songs, albums, and artists. However, we did not consider the fact that the complexity that came with many-to-many relationships would require us to implement SQL solutions for those relationships for both reviews and searches of songs, artists, and albums. Due to this reality, the database ended up being far more complex than initially intended, forcing us to edit the CREATE tables themselves various times before it would function as intended. For the Python section, there were many issues with functionality early because we were a bit unfamiliar with using Pycharm. But through lots of research and learning, we were able to get our program running. An additional issue that we encountered with our Python code was readability and modularity. We soon realized we needed to create functions and methods in our python code in order to make our code much more readable as we had so many procedures and functions we wanted to call, as we wanted to allow the users with as many options to interact with our application as possible.

3. *Realized or contemplated alternative design/approaches to the project*

Throughout the entire project, we definitely went back and forth changing a lot of parts. The UML diagram is something we went back and changed multiple times. We wanted to represent all the relationships needed given the actions we wanted the user to be able to make using our application. We added much more relationships than we thought we would need in the beginning. When we started working on writing the procedures, functions, and triggers, we realized that we needed to create more tables because we forgot to include some of the necessary tables for the user to be able to interact with the database in a way we mapped out in our flow chart. By going back and forth, we had to add so many new tables and it definitely expanded from the number of original tables

we had in our project proposal. Then, when it came to solidifying the procedures, functions, and triggers, we spent days just debugging them, as these procedures, functions, and triggers we created were much more complex than the ones we were used to from our homeworks and practices.

4. *Document any code not working in this section*

All of our code is working the way we want it to.

Future Work

1. *Planned uses of the database*

This application that we created can definitely be distributed to many different people, whether that be our friends or family. They can all interact with this database and be able to search and check based on what is already in the database as well as add, delete, and update. By allowing a lot of users to interact with our application, the information stored in it, especially the songs, albums, and artists as well as the reviews are extremely helpful for those looking for song, album, or artist suggestions for example. Our filters for searching are also especially useful if one is looking for suggestions. This application that we created is especially unique in the sense that all different streaming service users can interact with it and the fact that people can share as well as look for other people's thoughts on the collection of information we have.

2. *Potential areas for added functionality*

A potential area for added functionality is for users to input a suggested playlist of theirs given a description with the general type(s) of music they are interested in. This way, users on this application will not only be inspired to listen to certain songs, artists, or albums, but also other people's playlists that may contain a mixture of different artists for instance. But if a user has the same taste of music as the one who made the playlist, the user may be able to discover new favorites that way. Another functionality that can be added is allowing users to respond to other people's reviews. By allowing this, it can provide more user-to-user interaction. Also, an existing user will more likely continue to use our application as they will want to come back and see if anyone responded to their reviews.