

Proposal: Databasify

Project Team

Kaan Eroglu
Paul McDonald
Ryan Lisowski

Project Abstract

Spotify is a gatekeeper to incredibly useful data. However, the default interface does not allow users utilize this information at its fullest potential. Databasify is a web application that allows users to query information that cannot be reached by the average user. It uses a relational database that can access typical spotify data such as artists, albums, and songs. However, it also has the power to interact with “behind the scenes” data such as popularity, tempo, and even danceability. The web front-end will display data by executing database functions such as addition, minus, rename, strings, union, and joins. This application will allow Spotify users to query incredibly specific criteria and construct more flexible playlists with ease.

Project Scenario and Goals

Ideally, Databasify would be used to pull information from spotify in order to construct specialized playlists. The playlists are considered specialized because the application would allow users to use information collected from spotify’s API that is unreachable from the standard user interface. For example, a typical Spotify user is able to search for songs, artists, and albums by title and genre. However, in addition to this, Databasify would allow users to search by tempo, sort by popularity, filter explicit songs, and more.

Some examples of Databasify use cases may include:

1. Collecting the discography of an artist in chronological order of release.
2. Ranking all of an artist’s songs or albums in terms of popularity. Spotify only lets users see the top 10 songs by an artist.
3. Finding the most “danceable” songs by an artist.
4. Filtering for non-explicit or “clean” songs for the family.
5. Following all artists featured in specific playlists. Typically, a user can only follow one artist at a time by navigating to their page.

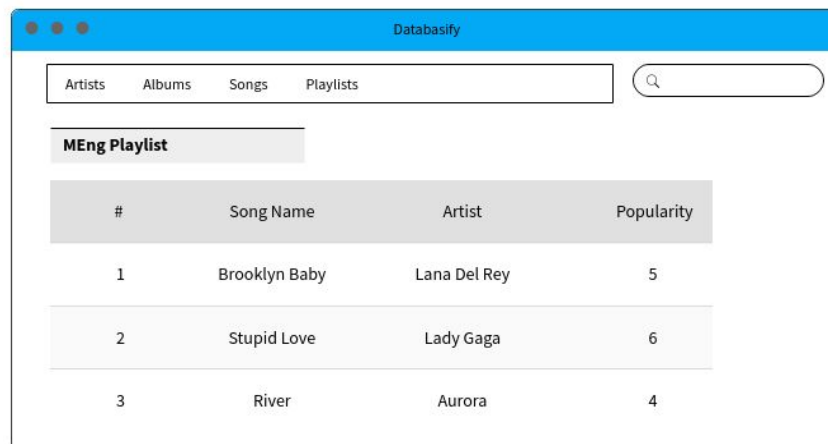
Given the time and technical limitations of this project, our goal is to establish a proof of concept for Databasify. To keep things simple, our database will only be populated with the work of a select few artists. Entries in the database will not be updated to reflect recent changes performed by Spotify. Instead, the focus will be on constructing an efficient and functional database. Since this is a proof of concept, the baseline goal is to have a populated, musical database capable of typical SQL queries. The implementation of all other features will be included only if there is sufficient time.

Design Strategy

The application will be built by using various technologies including;

- Python flask framework backend with HTML frontend and bootstrap styling
- Typescript for populating data on the frontend
- Sqlite for database
- SQLAlchemy for Object Relational Mapping
- Spotify API to populate data in the database and create playlists
- Creating JSON endpoints for possible API features

Wireframe for user interface:



#	Song Name	Artist	Popularity
1	Brooklyn Baby	Lana Del Rey	5
2	Stupid Love	Lady Gaga	6
3	River	Aurora	4

The application will be built by using agile methodologies. Initially, a simplified, functional version will be developed, and additional features/methods/queries will be added incrementally. Developers will have the ability to run queries directly on the database and inspect the results to see if they match the expected outcomes on the web interface. Github will be utilized for version control and as a collaborative work environment. Our design and development process will also include pair programming through Visual Studio Code LiveShare.

Design Unknowns and Risks

Since the project team has varied experience using databases, front-end web development, and interacting with APIs, several risks that have been identified. The first is that the team will need to be able to transfer Entity Relationship Diagram (ERD) concepts over to an actual database implementation. The relationships that have been shown in the diagram will need to be maintained within the database implementation. Ensuring that these table relationships are maintained will be critical to the project.

Data contained in the database will also have to be queried and shown on the front-end web page. Since the team has limited experience with front-end web development, properly displaying the song or album list information that has been pulled from the database will have some risks associated. These risks are mainly time-related since it will certainly be possible to display the information that should be shown on the website, but it may take time to determine how best to display that data.

Finally, interacting with the Spotify Developer API carries risk which includes both being able to get the information that is wanted, and not being cut off from that information by the company. Because we do not control the data, there is a chance that the team may not be able to extract as much of a volume of information as initially planned due to unforeseen API restrictions by the company. As well, the project team plans to be able to export a playlist generated by the team's application to a Spotify user account. From the team's initial research, this seems like a straightforward process, but there could be issues encountered during the implementation phase due to the use of a 3rd party API.

Implementation Plan and Schedule

Task ID	Task description	Deadline	Dependency
1	Investigate Spotify API and decide what we want to extract	3.10	-
2	Draw ERD Identify tables, columns, relationships	3.15	1
3	Relational Model	3.29	2
4	Functional Model	4.12	3
5	Identify mysqlite functions and page URLs	3.20	4

6	Create front-end templates	3.28	5
7	Create queries for each function	4.02	5
8	Link functions to front end templates	4.10	7
9	Final Report	4.15	8

In general, each of the tasks specified in the implementation table will be worked on with consistent effort once the dependencies for that task have been completed. The goal will be to have each of the tasks completed and in final group review at least four days prior to the deadline.

Each team member will have the opportunity to work on each of the tasks, including the database design and implementation, the front and back-end of the website, and the Spotify API utilization. The team will break up each task into approximately equal-sized subtasks that will be assigned to each member.

During task implementation and upon task completion, unit and integration testing will be planned and undertaken to ensure that the application continues to function as intended.

Evaluation

Several metrics, included in the table below, will be considered in order to evaluate whether the database and front-end application was successful.

Evaluation Metric	How to Display in the Final Report
Was the original database design adhered to?	Any differences between the original database design and the final implementation will be shown and discussed in the final report.
More than 80% of the material in class covered.	A table will be included in the final report that includes the material learned in class. This table will indicate which material has been included in the database implementation.
Was a custom playlist generated and shown on the website?	The core feature of this application is to have a playlist generated from the user's inputs, and to have this playlist displayed on the website. Screenshots from the website will be included in the

	report, along with a discussion.
(Optional) Was a custom playlist generated and added to Spotify?	An optional objective is for the team to create a playlist on the Spotify platform for the user by using Spotify's playlist API functionality. This will be discussed in the final report.

One of the main tradeoffs that will be evaluated is whether the database tables should be normalized to BCNF or not. Microsoft, for instance, included a discussion in an article titled "Description of the Database Normalization Basics"¹ about the extent to which databases should practically be normalized since having too many small tables could introduce performance problems. Performance issues are unlikely to be encountered with this project due to the limited size, but the project team will research and discuss whether issues could become apparent when the database is scaled up.

¹ "Description of the database normalization basics," 10-May-2017. [Online]. Available: <https://support.microsoft.com/en-us/help/283878/description-of-the-database-normalization-basics>. [Accessed: 06-Mar-2020].