**SpotiFlask**

Database project part 1 – design documentation

By James O’Grady



(<https://developer.spotify.com/>)



(<https://www.fullstackpython.com/flask.html>)

# Preliminary Analysis

My proposed system is named “SpotiFlask”. This system will be used to scrape and then store data surrounding a user’s Spotify listening habits and existing playlists, as well as any liked songs. This data is stored in a database powered by sqllite3, an inbuilt python SQL database library. The database will be accessible by the user via an online UI, built out of python and powered by an external library, “Flask”. This use of Spotify data and a Flask powered UI gives the name, “SpotiFlask”.

The reason for the systems existence in the first place is simple – it provides an easily accessible way to obtain and see data surrounding a user’s listening habits, favourite songs, data about those songs that is usually hidden, favourite artists and favourite genres of music. For the purposes of the project, the sample data to be analysed will be my own.

The system is highly feasible, as the Spotify API and all other external and internal modules used are free to use, meaning there are not any economic factors to consider when developing the database. The only operational factors to consider will be the time that the development of the UI and database. Technical factors that need to be taken into account include the skills necessary to develop , as well as any skills needed to work with the Spotify API and the Flask UI. I have these skills, as well as the time needed to develop the project, so both operational and technical factors are met.

## Gantt Chart

See attached document.

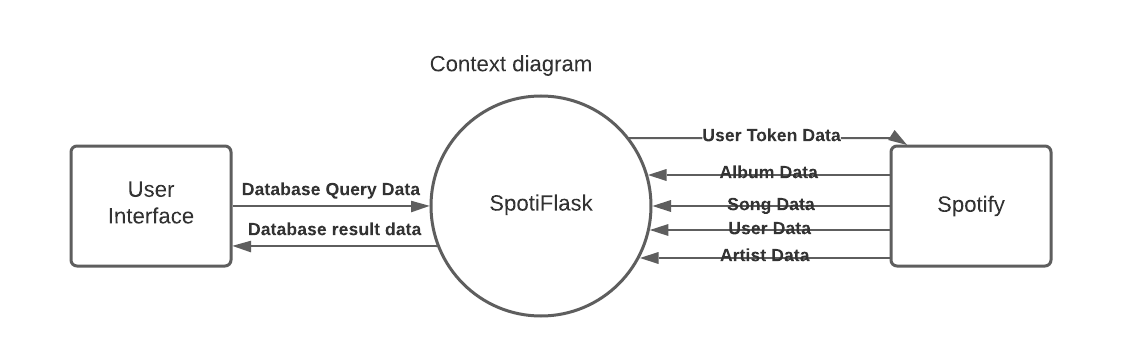
# Analysis

The database will be developed to be the data storage of a Spotify integration that obtains and analyses data around a user’s listening habits. The integration will provide access to usually hidden parts of Spotify, accessible only through the API, such as song popularity.

## Purpose

The purpose the database is going to serve is to store data surrounding songs, playlists, artists, albums and the user. The database will improve the system by allowing quick and easy access to data through use of SQL queries. The database being powered by SQL allows for the filtering of data and customisable queries to find any data as long as it exists.

## Context Diagram



## Requirements of the database

The database will need to be able to store large amount of long term data, as well as allow the addition of any new data and removal of any old data. The user should also be able to retrieve any data from the system through use of SQL queries built into the system.

## Ethical Considerations

There are a few ethical considerations around the database that need to be addressed. Firstly, the database should be able to securely and safely store a Spotify user token that can be used to obtain data within specific scopes of Spotify. Secondly, the database will be storing a whole lot of data around a user’s listening habits and this information could be useful to companies to build a profile of the user for advertising and marketing purposes. This data is protected by the Data Privacy act (1988) and 13 Privacy principles that came into effect in march of 2014. In keeping with these, the data is only accessible to the user who owns it and the system, and will not be disclosed to third parties and is kept secure through the use of a localised database accessible only through the SpotiFlask UI or through the local user, who should also be the owner or the data being protected.

# Design

## Data Dictionaries

### User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Length | Description | Primary / Foreign |
| Username | CHAR | 25 | A Spotify user’s unique identifier that Spotify uses to keep track of their account. | Primary |
| UserToken | VARCHAR | 250 | A Spotify token for use in accessing user data such as playlists, followed artists and liked songs. | N/A |
| IsPremium | BOOL | N/A | Whether the Spotify account is premium. | N/A |
| DisplayName | VARCHAR | 30 | The user’s display name on Spotify – what other users see. | N/A |

### Playlists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Length | Description | Primary / Foreign |
| PlaylistID | CHAR | 22 | The Internal Spotify ID of the playlist. | Primary |
| SongCount | INT | 10000 | The amount of songs within the playlist. | N/A |
| UserID | CHAR | 25 | The creators username | Foreign |
| PlaylistName | VARCHAR | 100 | The display name of the playlist | N/A |

### Songs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Length | Description | Primary / Foreign |
| SongID | CHAR | 22 | The Internal Spotify ID of the playlist | Primary |
| Duration | INT | 2147483647 | The length of the song in milliseconds | N/A |
| SongName | VARCHAR | 200 | The name of the song | N/A |
| Popularity | INT | 100 | How popular the song is (Calculated by Spotify internally) | N/A |
| Genre | VARCHAR | 20 | What genre the song is apart of | N/A |
| Album | CHAR | 22 | The ID of the album the song was released as a part of. | Foreign |
| Artist | CHAR | 22 | The artist id of the owner of the song | Foreign |

### Albums

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Length | Description | Primary / Foreign |
| AlbumID | CHAR | 22 | The internal Spotify ID used to identify the album | Primary |
| AlbumName | VARCHAR | 200 | The name of the album | N/A |
| AlbumSongs | INT | 10000 | How many songs are in the album | N/A |

### Artists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Length | Description | Primary / Foreign |
| ArtistID | CHAR | 22 | The internal Spotify ID used to identify the album | Primary |
| ArtistName | VARCHAR | 100 | The name of the artist | N/A |

## Many To Many Relationship Resolving Tables

### SongAlbums

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Length | Description | Primary / Foreign |
| SongID | CHAR | 22 | The Internal Spotify ID of the song | Foreign |
| AlbumID | CHAR | 22 | The Internal Spotify ID of the album | Foreign |

### SongArtists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Length | Description | Primary / Foreign |
| SongID | CHAR | 22 | The Internal Spotify ID of the song | Foreign |
| ArtistID | CHAR | 22 | The Internal Spotify ID of the artist | Foreign |

### PlaylistSongs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Length | Description | Primary / Foreign |
| PlaylistID | CHAR | 22 | The Internal Spotify ID of the playlist | Foreign |
| SongID | CHAR | 22 | The Internal Spotify ID of the song | Foreign |

### AlbumArtists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Data Type | Length | Description | Primary / Foreign |
| AlbumID | CHAR | 22 | The Internal Spotify ID of the album | Foreign |
| ArtistID | CHAR | 22 | The Internal Spotify ID of the artist | Foreign |

## User Interface

When creating my user interface, I have kept it simple, so as to make it easier for a new user to simply open the webpage and begin searching through data. The colour scheme is mostly black and white as there will not be any pictures that require colour. There is a page for creating and destroying the data base, a page for running queries on the database, a user guide for using the UI and a homepage to navigate around the UI, as well as listing the names of any tables.

### Homepage

Appendix 1.

### Tables

Appendix 2.

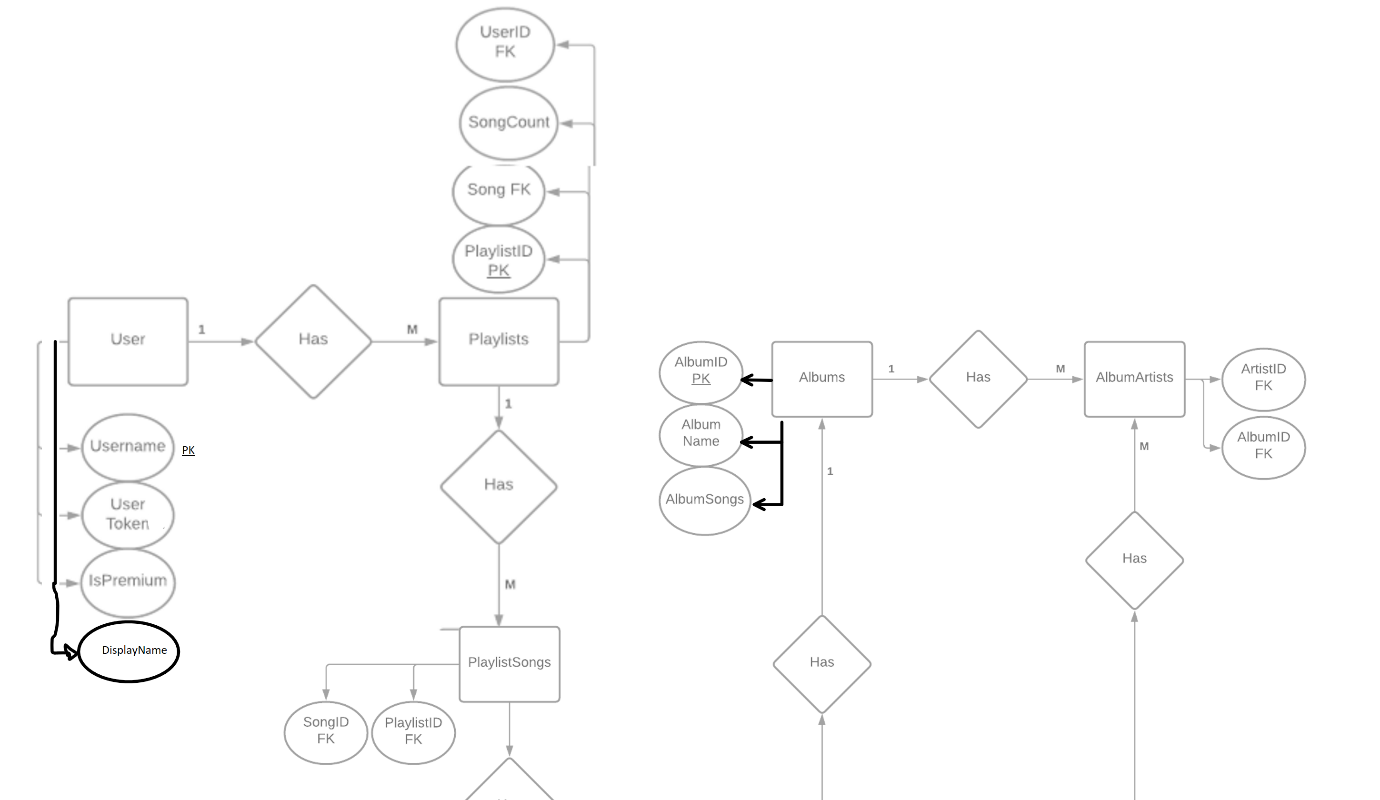
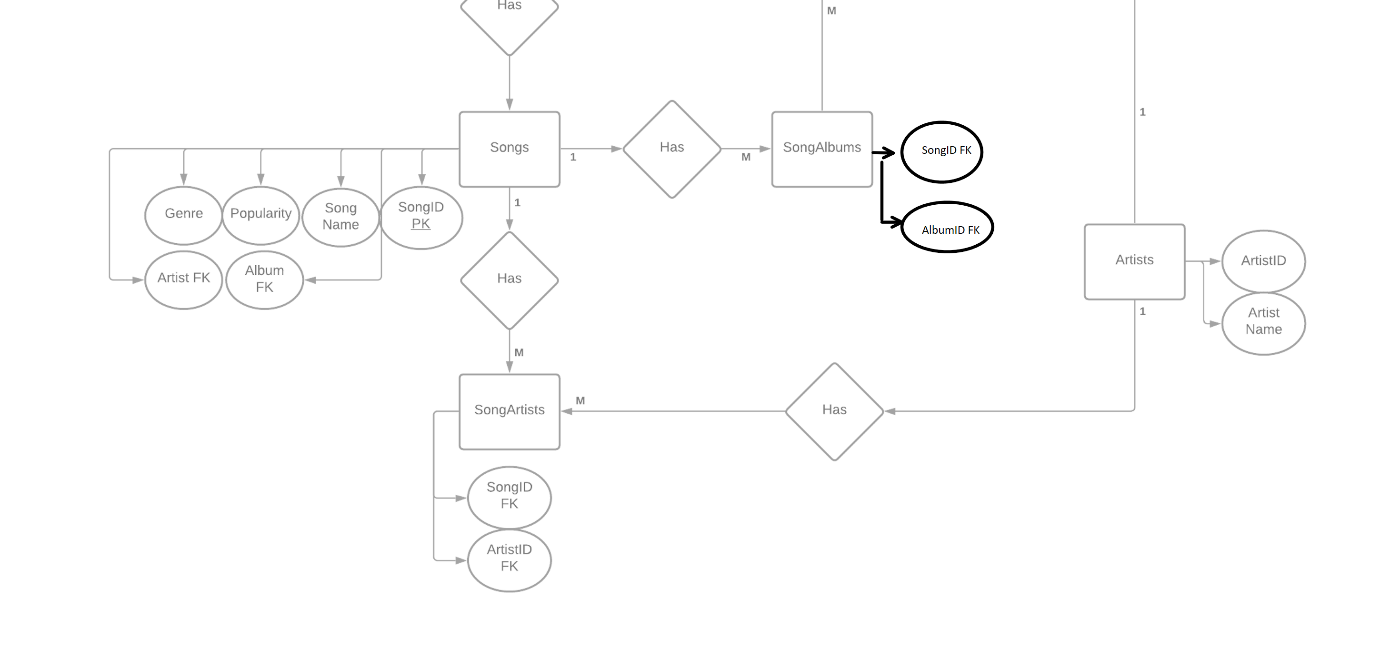
### Run Query

Appendix 3.

### User Guide

Appendix 4.

## Entity Relationship Diagram

Duration

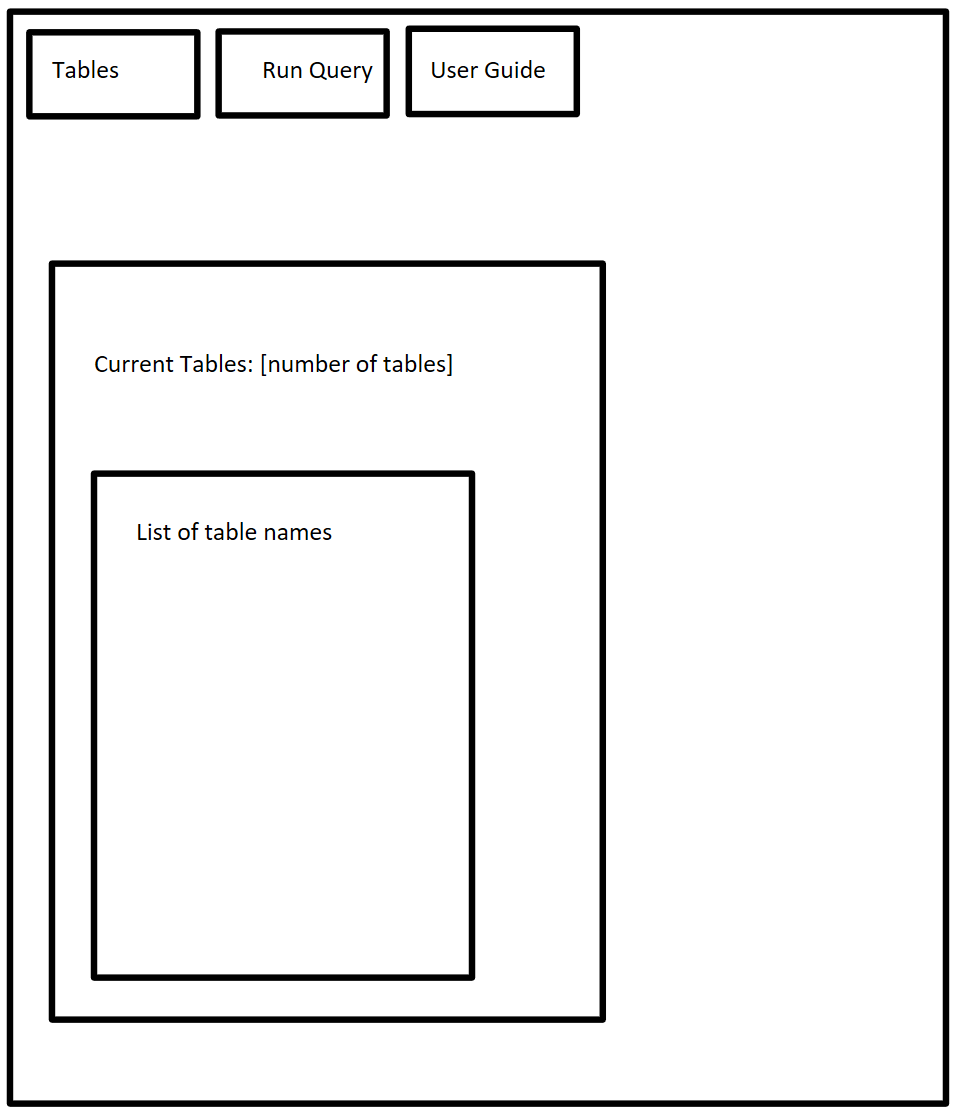


## Sample Queries

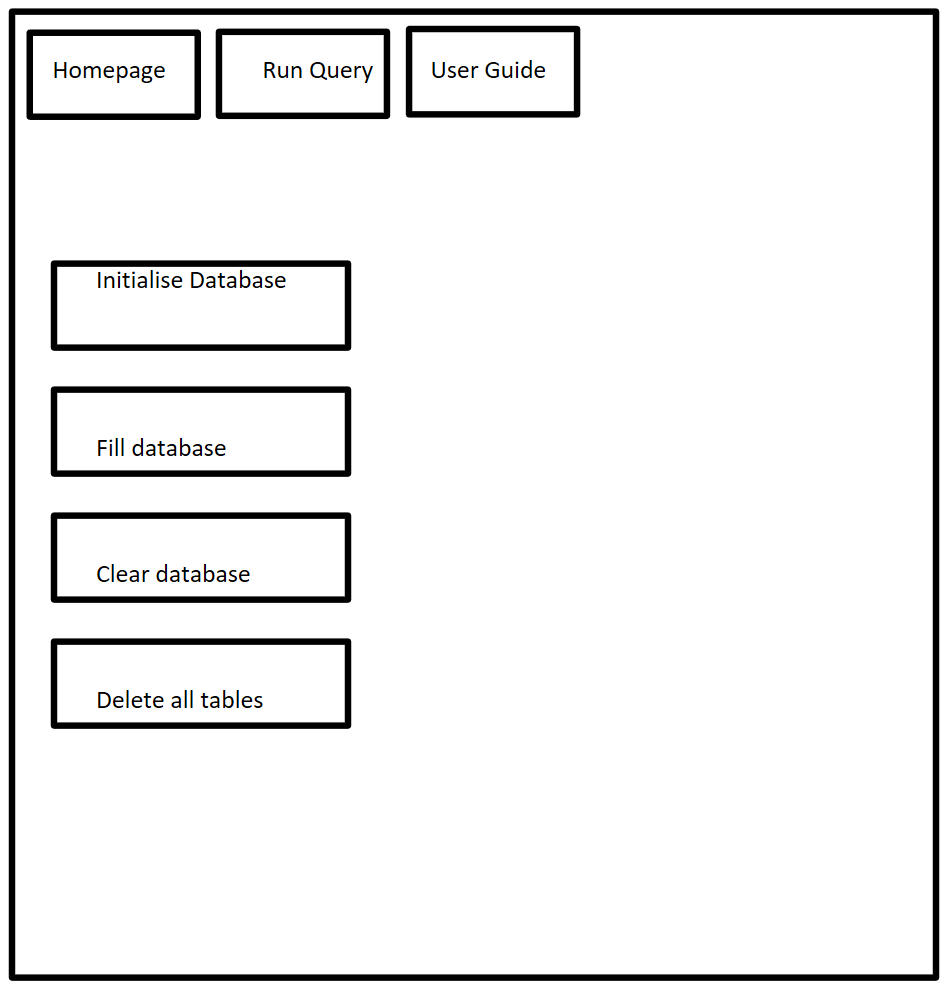
Some of the queries that the user needs to be able to perform on the database include being able to find a song by its name, by its artist, finding all the songs in a playlist or album and finding all the songs with a popularity between 2 values, or greater then or less then a value. The user should also be able to find the owner of an album or playlist, its name and how many songs are within it. Apart from this, the user should be able to print out all the records in a specified table, or multiple tables.

# Appendixes

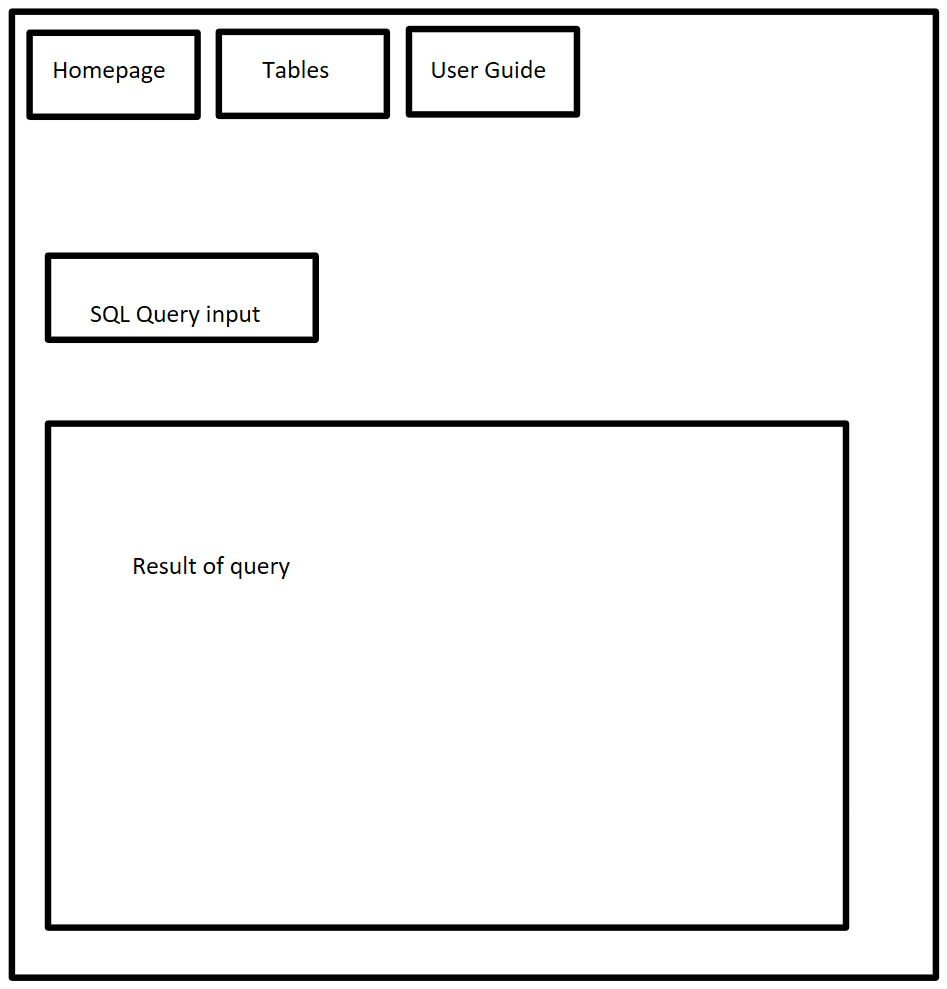
## Homepage for UI Design



## Tables page for UI design



## Run Query page for UI design



## User Guide page for UI design

