Team 5:

Rebecca Arce

Tyler Correll

Justin Ingwersen

Ali Kinay

Team Assignment 5

1. **Cache**

Will be calling Spotify API to get songs from all existing user playlists and store into cache. This will happen every time a user logs in on Spotify through our app - which will only happen when the someone uses our app for the first time. If the user wants to update their playlist songs they may sign out of Spotify and re-login to get an updated list of songs. For that case scenario we will have a “refresh” option where the user would automatically be logged out and logged back in. Hence, we will only be calling the Spotify API once if the user does not choose to refresh their playlists.

1. **Data Documentation**

We will be using MySQL because we are more familiar with the syntax and overall functionality of SQL. We will have 4 tables. The first 3 will all be connected by the column user\_id, which is an encrypted number that will be given to each username.

Table **User** will be a private table that stores username and an encrypted user\_id:

* Columns: user\_id, username
* Primary key: user\_id

Table **User\_Info** will be a public table that stores some basic user information:

* Columns: user\_id, location, age
* Primary key: user\_id

Table **Playlists** will be a public table that have playlist names per user:

* Columns: user\_id, playlist\_name
* Primary key: user\_id

Table **Songs** will be a public table will have songs inside each playlist:

* Columns: playlist\_name, song
* Primary key: playlist\_name

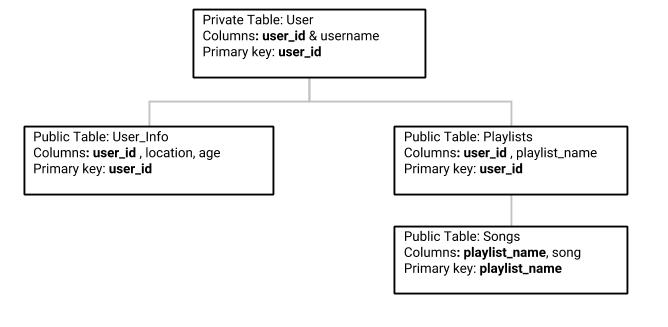


Figure 1: ERD for SQL tables. The columns in bold are each tables’ primary key.

In order to preserve security we are going to associate each username with an encrypted user\_id. This will allow us to still use a relational database yet keep the sensitive information private. We are also planning on taking an extra measure of making the user table private to limit access. Our main table is going to be the “User”. Once the individuals sign in with using Spotify API (oAuth) they will be presented with the main page of our website. We will store their user\_name, create their encrypted user\_id and use this number to fetch information in other tables and display their specific playlists. Songs will be connected to playlists in the songs table.

MySQL Syntax for the Database

CREATE DATABASE tune\_active\_database

|  |  |
| --- | --- |
| \*private table.  CREATE TABLE User (  username varchar(255),  user\_id int NOT NULL,  PRIMARY KEY(user\_id)  ) | CREATE TABLE User\_Info (  user\_idint NOT NULL,  location FLOAT,  age int,  PRIMARY KEY(user\_id)  ) |
| CREATE TABLE Playlists (  user\_idint NOT NULL,  playlist\_name varchar(255)  PRIMARY KEY(user\_id)  ) | CREATE TABLE Songs (  playlist\_name varchar(255),  Song varchar(255),  PRIMARY KEY(playlist\_name) |

1. **Sequence Diagram**

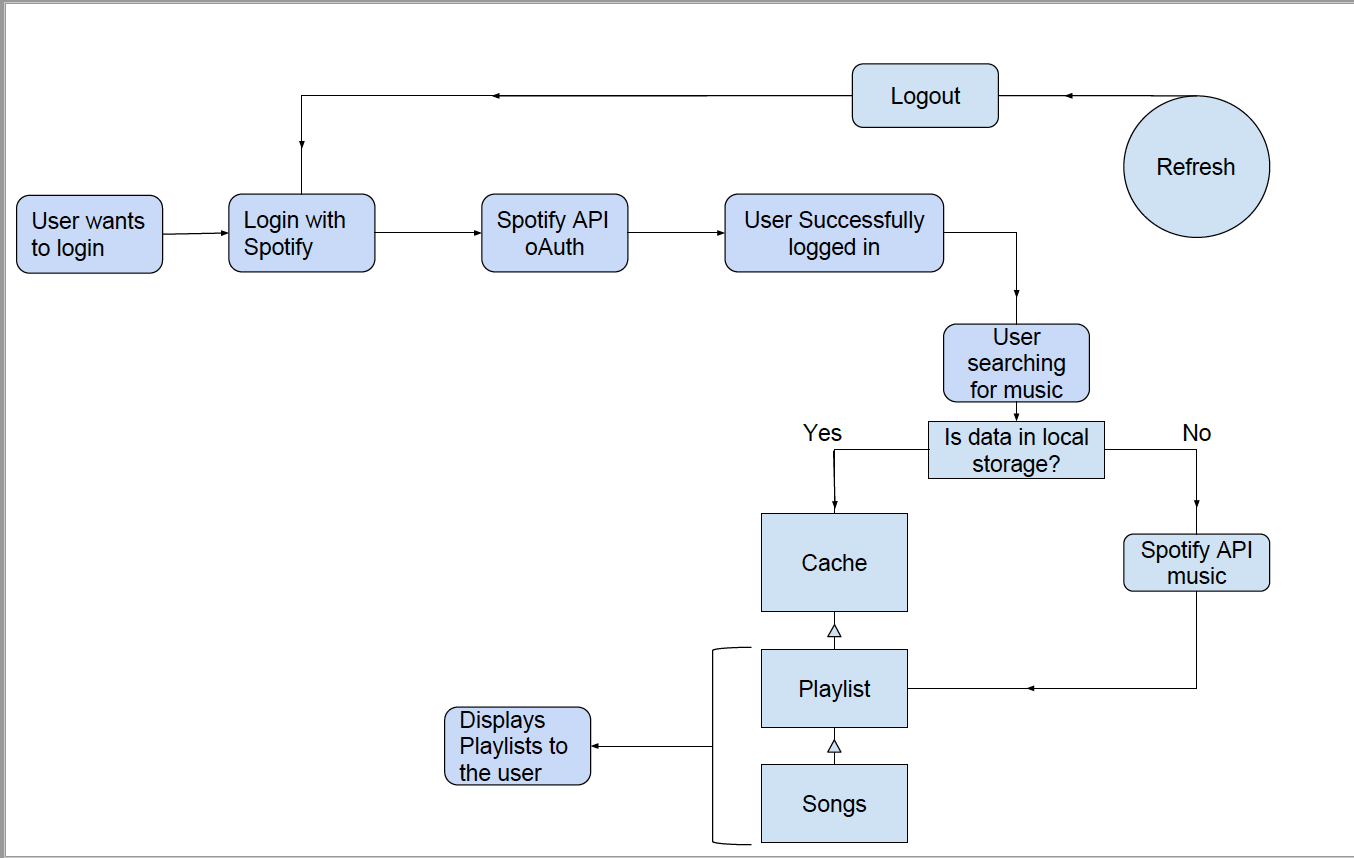
****

Figure 2: Sequence diagram for our cache - beginning at the farmost left. The “refresh” feature will be implemented as a means to update user playlists. This will be done by calling the Spotify API again by quickly logging out and logging back in.