Vibechek

Robert Babaev 101143382

COMP 3005 W22 Final Project

Table Of Contents

Purpose and Description of the Database

Vibechek is intended as a music scheduling service that allows users to map out their Spotify playlists in schedules spanning a week's time, and seamlessly switch between playlists without having to fiddle with the Spotify player. This is so that a user can reduce the amount of time wasted scrolling through Spotify looking for songs and/or playlists.

The main functionality of the database is the scheduler. Schedules are subdivided by weekday, and weekdays are subdivided into blocks, which each have one playlist that they play. Weekdays correspond to days of the week, and block slots are 30 minute blocks of time, where blocks can take up multiple block slots in a day. The user should be able to manipulate these schedules as they see fit, and can even save schedules from other users for their own use. Users also have the option to make their accounts and schedules private if they so choose.

Ideally, users would first register with the application and allow the associated Spotify application access to their account. This way, they can begin saving their playlists from Spotify, and marking themselves as the owner of that playlist. Once a user has some playlists saved, they can create blocks with durations in increments of 30 minutes. These blocks can be placed throughout a Vibe Day at different start times, and the same block can be reused multiple times, even within the same day. The same goes for Vibe Days within schedules, since a user can equip the same day or different days in multiple weekdays of their schedule, and even across schedules. As far as scheduling goes, it's as flexible as can be with increments of 30 minutes.

All scheduling is designed to be done in relation to the active user, so the app also stores the user's timezone. In future iterations, email verification would be performed on registration for added security, as well as for the ability to reset a user's password, which is, unfortunately due to time constraints, currently impossible. Additionally, since the scheduler uses the Spotify API, tokens and expiry times are necessary to access the API, as well as avoiding too many requests to the API itself in the event of token expiry (prerelease access has an unspecified rate limit which I have to assume is lower than any remotely acceptable production rate limiting).

ER Model