Team: Dewbed

Members: William, Bettina, Emily, and Devika

Project: 2A - Preliminary Architecture

Preliminary Architecture:

https://docs.google.com/document/d/1wc58_iXdFFnJTfAqEHvy1596dyz1_m5cbDPMeigivDE/edit?usp=sharing

Primary Author: Devika

Document Table of Contents

I. **How Introspective Spotify Works**

II. **Getting Started Commands**

III. **Music Theory Commands**

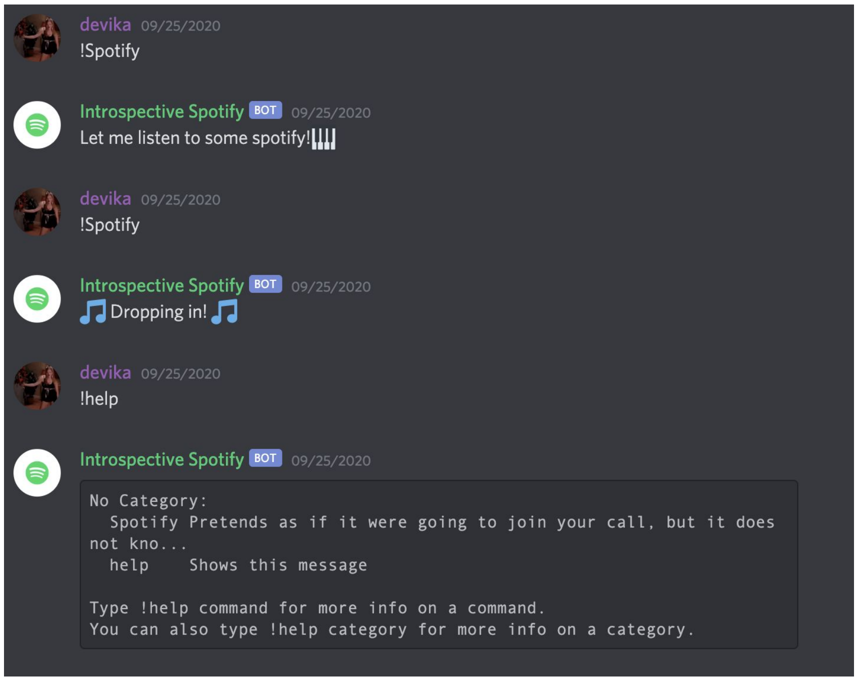
IV. **Music History Commands**

V. **Synchronous Listening Commands**

VI. **Spotify Wrapped Commands**

I. How introspective Spotify works:

- Text in discord chat to interact with the bot. We conducted a prototype which interacts with dummy messages such as !Spotify to get the bot to respond. Using Python, the bot has specific bot commands and a !help guide. Similarly, the bot will also send a message to any new users and can react to specific messages (currently edited out due to spam).



devika 09/25/2020
!Spotify

Introspective Spotify BOT 09/25/2020
Let me listen to some spotify!🎹

devika 09/25/2020
!Spotify

Introspective Spotify BOT 09/25/2020
🎵Dropping in!🎵

devika 09/25/2020
!help

Introspective Spotify BOT 09/25/2020

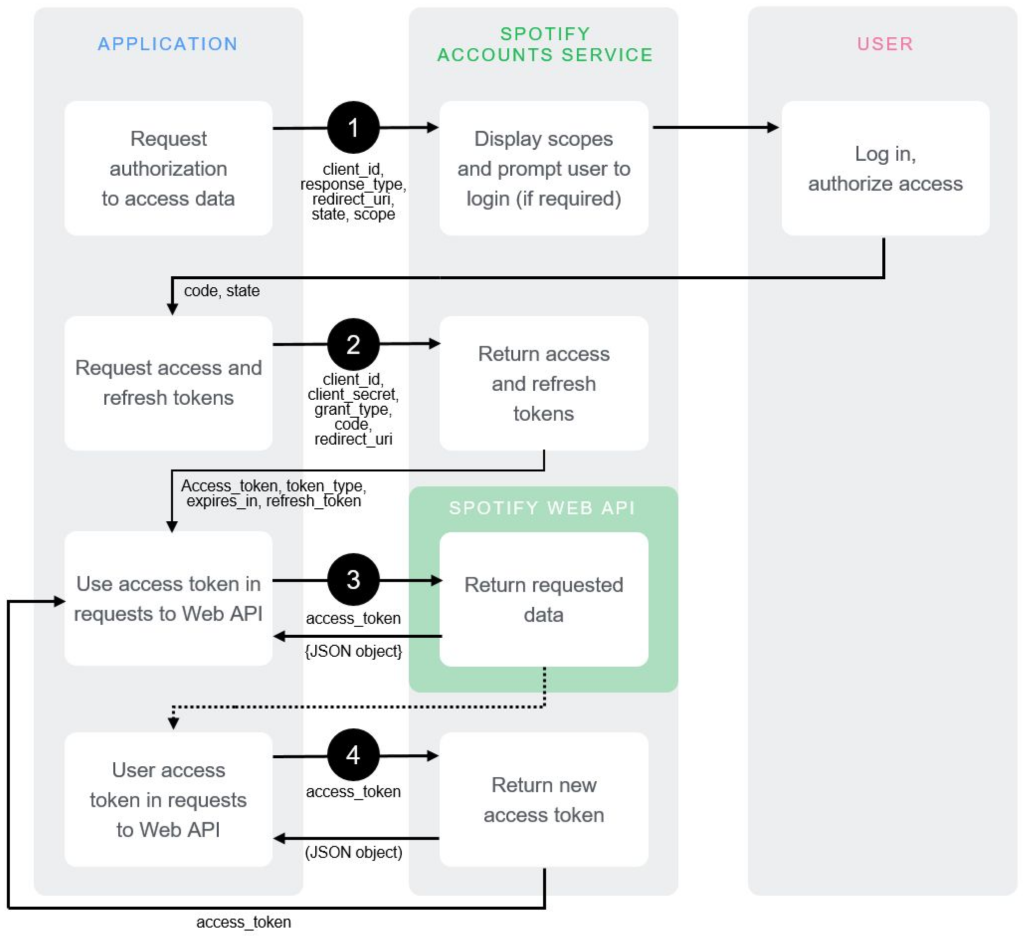
```
No Category:
  Spotify Pretends as if it were going to join your call, but it does
not kno...
  help    Shows this message

Type !help command for more info on a command.
You can also type !help category for more info on a category.
```

- Music plays on your own Spotify account. Currently, it is possible to play music on Spotify through Discord through listening parties in a channel, and listening along on individual Spotify.

II. Getting Started Commands:

- **Overview:** Type these commands in the Discord chat to get started! These commands also aim to increase usability.
- **About the Commands:**
    - **!Help:** Sends the discord channel a list of all of the discord bot's commands.
        - If users type **!Help [command]**, will provide a detailed description of the specific command
    - **!Login:** Sends the user a direct message asking them to log into their Spotify account via Introspective Spotify and to give permission for Introspective Spotify to modify their data. This will be accomplished by interacting with Spotify Web API and OAuth2.0 via HTTP requests to provide the user with an external login screen and permissions acceptance page.
    - **!Logout:** Allows the user to log out of Introspective Spotify and revoke permissions to the user's Spotify account.



APPLICATION
SPOTIFY ACCOUNTS SERVICE
USER
Request authorization to access data
1
client_id, response_type, redirect_uri, state, scope
Display scopes and prompt user to login (if required)
Log in, authorize access
code, state
Request access and refresh tokens
2
client_id, client_secret, grant_type, code, redirect_uri
Return access and refresh tokens
Access_token, token_type, expires_in, refresh_token
SPOTIFY WEB API
Use access token in requests to Web API
3
access_token
{JSON object}
Return requested data
User access token in requests to Web API
4
access_token
(JSON object)
Return new access token
access_token

source: https://developer.spotify.com/documentation/general/guides/authorization-guide/

III. Music Theory Commands:

- **Overview:** Music Theory Commands provides the user music theory analysis about a song.
- **About the commands:**
    - **How it works:** All music theory commands use the Spotify API to retrieve song IDs (accessed via standard HTTPS requests in UTF-8 format to an API endpoint). Use the song ID and access token (access tokens are granted after user logs-in. See diagram above) to request from "Audio Features" documentation. Introspective Spotify bot will respond in the server chat with the requested music theory information.
        - **[song]:** this parameter has type song and refers to the search query keyword used to search Spotify's API. If no song is provided, this parameter will find the current song.
    - **!Key [song]:** Use "Audio Features" documentation to get the estimated key of the track.
    - **!Tempo [song]:** Use "Audio Features" documentation to get the estimated tempo (beats per minute) of the track.
    - **!TimeSignature [song]:** Use "Audio Features" documentation to get the estimated time signature of the track.
    - **!Mode [song]:** Use "Audio Features" documentation to get the estimated mode (major or minor) of the track.
    - **!Mood [song]:** Use "Audio Features" documentation to get the estimated mood (happy, sad, etc.) of the track.
    - **!Danceability [song]:** Use "Audio Features" documentation to get the estimated danceability (high, medium, low) of the track.
    - **!Acousticness [song]:** Use "Audio Features" documentation to get the estimated acousticness (high, medium, low) of the track.
    - **!Energy [song]:** Use "Audio Features" documentation to get the estimated mode energy (high, medium low) of the track
    - **!Instrumentalness [song]:** Use "Audio Features" documentation to get the estimated instrumental (high, medium, low) such as "Oohs" and "Aahs" in a track.
    - **!MusicTheoryHelp:** This command gives a description of key, tempo, time signature, mode, mood, danceability, acousticness, energy, and instrumentalness for users that do not have familiarity with these terms. We will type these descriptions ourselves and then send a private message to the requested user explaining what each theory term means. We will ensure it is simple and easy to read while using emojis as well to make it look friendly.

IV. Music History Commands:

- **Overview:** Music History Commands provides music theory analysis over a span of time that they listened to music on Spotify. The response will be a bot message with the requested information.
- **About the commands:**
    - **!Genre [timeframe]**: tells users their most listened to genre of music during a timeframe. To obtain the genre, this command will get the top 10 artists using Spotify Web API's audio feature which returns an object that contains the associated genre of an artist. The command will then analyze the genres given and tell the user what genre is the most common genre in the user's top 10 artists.
    - **!TopSongs [time_range] [limit]**: tells user what their most listened to songs are.
        - **Parameter Descriptions:**
            - **[time_range]:** this parameter indicates the time range of the user's history to retrieve data. long_term (several years of data and including all new data as it becomes available), medium_term (last 6 months), short_term (last 4 weeks). Default: medium_term. Taken from Get a User's Top Artists and Tracks.
            - **[limit]:** this parameter refers to the number of top tracks the user wants to get with a minimum of 1 and a maximum of 50. If omitted, will default to the top 5 tracks/artists.
        - **Reaction:** After giving the users their top tracks, the Bot can ask the users if they want a more detailed analysis of their top tracks. Users will react to the bot's analyze message if they want to learn more. Discord Bot API for interpreting reactions. The Spotify id tracks will be taken and be analysed using Spotify Web API's get audio feature for several tracks. The tracks will be analysed individually and given as an object whose key is audio_features and whose value is an array of audio features objects pertaining to the tracks in JSON objects. Based on the array of features given, the analyse command will take the most prevalent key, tempo, time signature and mode of all tracks.
    - **!TopArtists [time_range] [limit]:** tells user what their most listened to artists are.
        - **Parameter Descriptions:**
            - **[time_range]:** same as !TopSongs's time_range parameter
            - **[limit]:** same as !TopSongs's limit parameter but for artists instead of tracks
- **Music History Commands VS Music Theory Commands:**
    - The Music History Analysis is similar to Music Theory Analysis. The difference between these two commands is the music theory command analyses a certain song and music history command analyses multiple tracks. There is no !TopArtist reaction analysis because Spotify does not give much artist information besides the genre, which we already have the command for.

V. Synchronous Listening Commands:

- **Overview:** These commands are entered in the Discord chat to allow users in the same Discord server to sync their Spotify accounts. Music will be played on each user's Premium Spotify account. Using Discord's API for reactions, the bot will react to the user's messages to ensure that it has processed the command.
- **About the commands:**
    - **!Join:** If there is no current listening party, starts a listening party session by connecting to Spotify Connect Web API and adds the user to the listening party. If a listening party is already in session, this command simply adds the user listening party. Once users are in the listening party, they will be able to use any of the following commands.
    - **!Leave:** Removes the user from the listening party. If the user is the last person in the listening party, we close the listening party by disconnecting from the Connect Web API service.
    - **!Play [song name]:** searches for the specified song in Spotify and adds it to the Spotify queue of all users currently in the listening party using the Spotify queue endpoint. Will store the current listen party's played songs in local memory until the listen party is over.
    - **!Queue [song name]:** Displays all songs added in the listening party's queue.
    - **!Pause:** Pauses the playback for all users in the listening party.
    - **!Skip:** Skips current song.
    - **!Rewind:** Plays previous song.
    - **!Shuffle:** Shuffles the songs on users' Spotify queues. We can not use the Spotify API shuffle endpoint as we have to ensure the same order of songs for all users in the listening party. We will shuffle the locally stored queue and then call the remove/add API endpoints to remove the old list of songs and add the newly ordered list of songs to all of the users' queues.
    - **!CreatePlaylist:** Creates a playlist of all the played and queued songs for the current listen party. The playlist will be created on the user's Spotify account and a link to the playlist will be provided by Introspective Spotify. All of this is done through Introspective Spotify interacting with Spotify Playlist API.

VI. Spotify Wrapped (Music History) with Friends Commands:

- **Overview**: These commands will be entered in the Discord chat to allow users in the same Discord server to either do a server wide analysis of overall music history or a partner analysis. The response will be a bot message @ing either everyone or the two accounts being analyzed.
- **About the commands:**
  - **!ServerAnalyze [timeframe]** - Creates a Spotify Wrapped (runs music history commands) for the whole server over a certain time period. Will provide average analytics for BPM, genre, and key, and reply with a message containing the information. The specific **[timeframe]** would only be "short, medium, and long" due to Spotify's limitations. Using Spotify's API specifically made for top tracks and "Audio Features" documentation, we can easily pool user specific audio features and top tracks and average the numbers out using our own algorithm (the most listened to genre being the average, using averaging to find BPM, and most listened to tracks becoming average) to display back in a simple Bot reply message formatted so it is easy for all users to glance over.
  - **!FriendAnalyze [username] [timeframe]:** Creates a personalized Spotify Wrapped (runs music history commands) between requesting user and another **[username**] in the server between a **[timeframe]**. Will provide average analytics for BPM, genre, and key between the two users and will reply with a message containing the information. The specific timeframe would only be "short, medium, and long" due to Spotify's limitations. Using Spotify's API specifically made for top tracks and "Audio Features" documentation, we can easily pool user specific audio features and top tracks and average the numbers out using our own algorithm (the most listened to genre being the average, using averaging to find BPM, and most listened to tracks becoming average) to display back in a simple Bot reply message formatted so it is easy for all users to glance over.