Brief introduction

RacTrack (short for Raccoon Track) is a music streaming interface designed to work in conjunction with Discord (a free online chat service similar to Slack). RacTrack will take in commands from any of the text channels in the Discord server, play audio in the voice channels of the server, and return information about the song in the text channel below the command (ie, if someone requested a Beatles song, it would play the song and then send a message with a link to the song, the date it was made, like/dislike ratio, and number of views).

Define Problem

When playing video games, some users prefer to listen to music different from the in game soundtrack. When they're playing with friends, it is difficult if not impossible to sync up music between 2 or more people.

Use scenario

**Case 1**

Start: Aidan wants to play a song on his Discord server that his friends can listen to.

Events: Aidan will use the search function on RacTrack to look up a specific song.  
Problems:

* Search result is done correct, but the song Aidan was looking for wasn't the top result (which got played), but the second or third result.
* Search turns up no results, leading to no song being played
* Search returns a non-valid source (Ie, a YouTube channel or playlist instead of video)
* Song quality is low, or distorted due to lag.

Concurrency:

While Aidan is playing music, other song requests from the server are thrown into a queue, which contains the song to be played, as well as who requested the song. Song requests from other servers feature their own queue are unaffected.

Finished state:

Aidan and his friends hear RacTrack join their voice channel, and the requested song starts playing. While this happens, RacTrack sends a message to a Text Channel containing information about the song.

**Case 2**

Events: A group of friends are chatting over discord. They are looking for a laugh between friends, and have RacTrack on the server. They will enter the command .racplay <song\_name>. RacTrack will then scour YouTube using certain search terms and filtering methods, then sets the quality of the ensuing video to 144p, to ensure the most laughable and absurd quality possible. The bot will play the song through voice chat and links it in the text channel, as well as posting analytics.

Problems: RacTrack may not find accurate results. Concurrency: While RacTrack is executing a command, it cannot be used by other members.

Finish state: A link to a poor quality video as well as poorly formatted statistical analysis image is posted to the chat, as well as the video being played over voice.

**Case 3**

Start: Matt wants to be able to play music to set the tone in his D&D session hosted online.

Events: He can search different styles of music using RacTrack to find the correct song to current environment.

Problems: If something is miss spelled, it can cause a wrong song to play. Songs can be full soundtracks which can be multiple hours or short 1-2 minute songs which then he might have to pick another song. If the server has no internet, or the internet is slow the song could not play or be very low quality.

Concurrency: Some users might not like the music and could skip the song if they don’t like it and could put on their own songs that might not fit the game. Finish state: A long soundtrack is playing in the background of Matt's D&D game.

**Case 4**Start: Scar wants to host a free radio station online but doesn't have enough money to pay for a website.

Events: He can ask RacTrack to play music for him and queue up songs.

Problems: - Scar might accidently play a live version or a lower quality version of a song. - Scar could have other people skipping or adding songs depending on if the bot is not setup correctly.

Finish state: A great online radio station hosted over a program many people already have installed and love to use

**Case 5**

Start: Tim found this awesome song by ACDC online, and wishes to show his friends how hip he is.

Events: Tim will type in the appropriate command for the software to look up his song, then it will play through the audio on the voice call.

Problems: - ACDC is not as good of a band as the scorpions. - He misspelled the song and he played Thunderstruck instead of back in black. - The version he got was the live version of the song, which is less quality than the studio recorded version. - Perhaps the voice call disconnected or his command didn’t go through.

Concurrency: Tim puts on the music, but his friend Fred doesn’t like the song and he can skip the song using a skip over command, and play his favorite song by Led Zepplin Kashmir.

Functional requirements:

Take user input from the chat log if and only if a command is typed.

* stream media from database into the voice-channel of the server, where the user who entered the command is.
* take information from database and send a message with the information in the chat.

example commands may include:

* play <songTitle>: seek a piece of media with the title provided and stream it to the voice chat, or add it to the queue.
* play <mediaURL>: stream the media from the link provided to the voice chat, or add it to the queue.
* pause: pauses the media currently playing.
* stop: stops playback of current media, removes it from queue and plays next song.
* queue list: list songs waiting in the queue.
* queue clear: remove all songs waiting in queue.
* RacTrack <songTitle>: like "play", but seeks the matching media with the lowest view count possible.

Non-functional requirements:

* software will run on a computer "host" which will then be accessed from users via a shared Discord server
* Media must be streamed without buffering or errors.
* must ignore irrelevant messages in chat that aren't commands
* Information about the media being streamed must include a source link, date of creation, number of listens/views, as well as the ratio of likes to dislikes (if applicable).
* Only play one piece of media at a time. Other requests will be put in a queue.
* if no media is found matching a search request, send an error message in chat.

System limitations

The RacTrack program will require a host computer running the program, after the software is running it can be accessed via any computer either running the Discord application, or via a web browser connected to the Discord server. The Program requires an internet connection on both machines to operate successfully, as well as the discord server to be operational.

Development Environment

RacTrack will be developed using Javascript, in conjunction with MongoDB and Discord.js API's and other system Libraries.

Deliverables

- Discord bot can read messages from server and send an appropriate response in text.

- Bot can recognize difference between a search-play request, and a play-link request

- Discord bot is able to hop into voice channel where user is and play a hard-coded sound (ie, a soundboard effect)

- Bot is able to play video in voice chat from a direct link

- Bot returns info on video being played in text (such as views, like/dislikes, upload date, etc.)

- Bot is able to play video in chat, given a title instead of a video link