Figure 1: The `maestro-cli` icon, which I made myself in Linearity Curve (inspired by MacOS's Terminal icon).

Diving into the listen-along streaming feature in

`maestro-cli`: A command-line music app

https://github.com/PrajwalVandana/maestro-cli

1. What it is

If a user is logged in as `praj`, they can stream their music on `maestro-music.vercel.app/listen-along/praj`. This is how it looks on the listener(s)'s end:

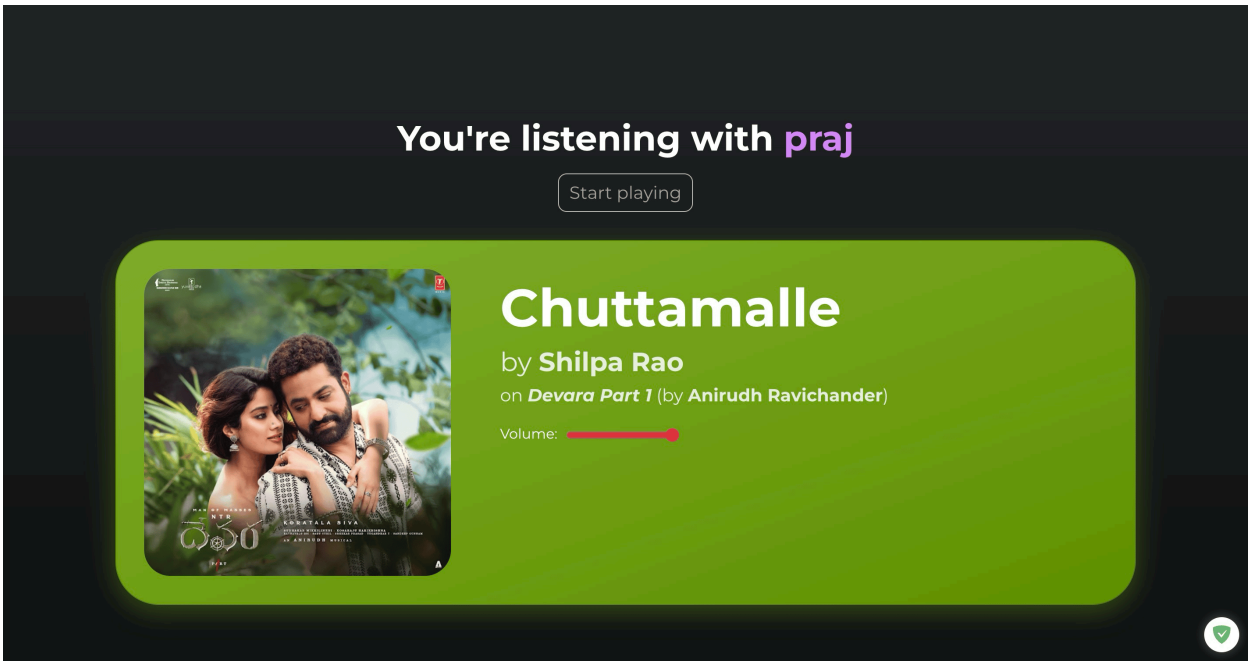

Figure 2: What a listener sees on `maestro-music.vercel.app/listen-along/praj`

2. What we need to make this work

1. A login system
2. A way to stream the music from the user's device to the website
3. A way to send the metadata (album art, song name, artist, etc.) to the website

3. How it works

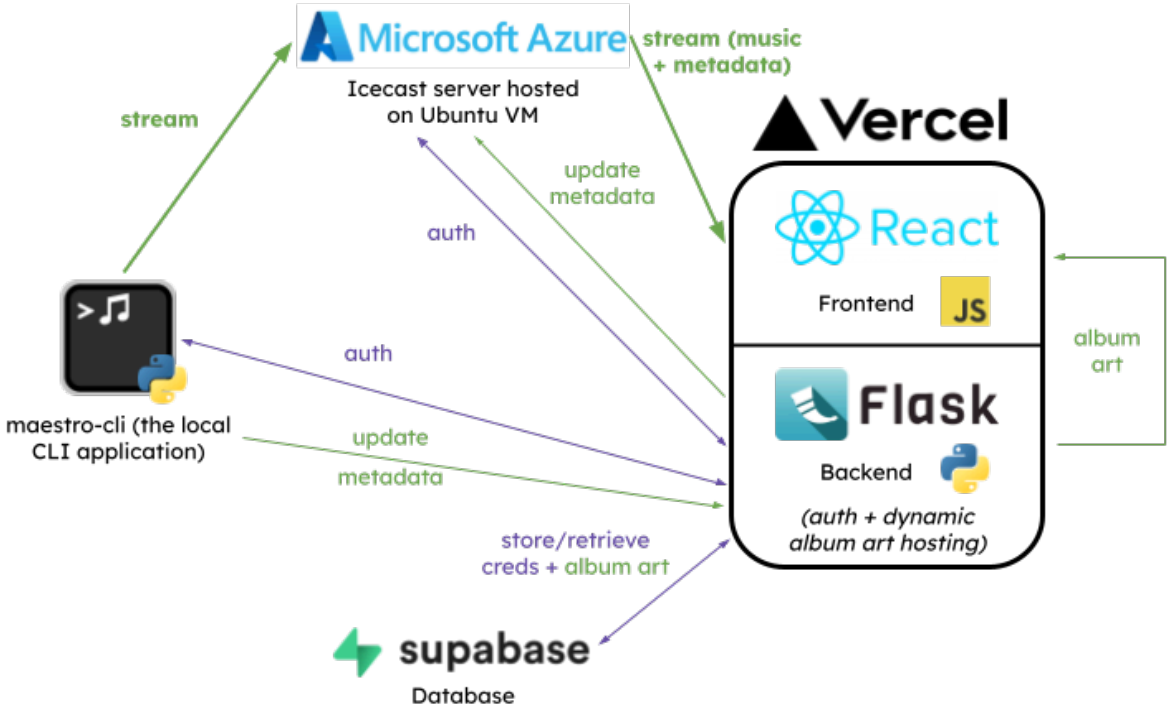

Figure 3: A diagram depicting the architecture of the listen-along feature.

1. A user (who we'll call the "player") creates an account using `maestro signup`. Let's say the username is `praj`.
   - This sends a POST request to the backend with the requested username and password.
     - If the username is already taken, the backend responds with an error.
     - Otherwise, the backend creates a new user in the database. No passwords can be stored in the database, for obvious security reasons, so only a hashed version of the password is stored.
2. The player begins a listen-along session using either `maestro play -S` or clicking `s` while playing music.
   a. A request (with credentials) is sent to the Icecast server to create a stream connection on the mountpoint `/praj`.
      i. The Icecast server sends a GET request to the backend to verify the credentials.
         - The backend checks if the credentials exists in the database by hashing the password and comparing it with the stored hash.

    ii. If the credentials are valid, the backend responds with a 200 status code, and the Icecast server creates the stream connection.
  b. Every time a new song plays (or when the song is paused; pausing is sent as metadata to the frontend to be displayed to listeners), `maestro` sends a POST request to the backend with the metadata of the song. The album art is encoded as a base64 string.
    i. The backend verifies the credentials and sends a GET request to the Icecast server (all metadata is converted to the form `"key1:value1|key2:value2..."` for compatibility with Icecast), replacing the player's credentials with the admin credentials (Icecast only allows metadata updates using admin credentials). The album art is not sent in this request, however, and is handled separately.
    ii. The album art's base64 string is stored in the database (replacing any previous album art for the player).
3. A user (who we'll call the "listener"; there can be an arbitrary number of listeners) navigates to `maestro-music.vercel.app/listen-along/praj`.
  a. The frontend attempts to connect to the Icecast mountpoint `/praj`.
    - If the player is not streaming, the frontend displays a message saying so.
  b. Icecast interleaves metadata directly into the music stream at regular intervals (usually 16000 bytes, this value is sent in the response headers when the connection succeeds), which the frontend parses and displays. The frontend also plays the music, obviously.
  c. Everytime the metadata changes, the frontend sends a GET request to the backend to fetch the album art.
    i. The backend retrieves the album art from the database and responds with it to the frontend.
    ii. The frontend displays the album art (the theme of the website changes to match the album art's color palette) and updates the metadata.
    iii. This API endpoint is also used to show the album art in the player's Discord status, if they have enabled that feature.