Project Report: Interactive Spotify Dashboard Using Node-RED.

# BACKGROUND AND MOTIVATION

Millions of people around the world use music streaming platforms like Spotify where they can listen to all their favourite tracks while their analytics are silently tracked in the background. For users to access these analytics they would need to wait until the end of the year for “Spotify Wrapped” which summarizes their listening habits for the past 12 months.

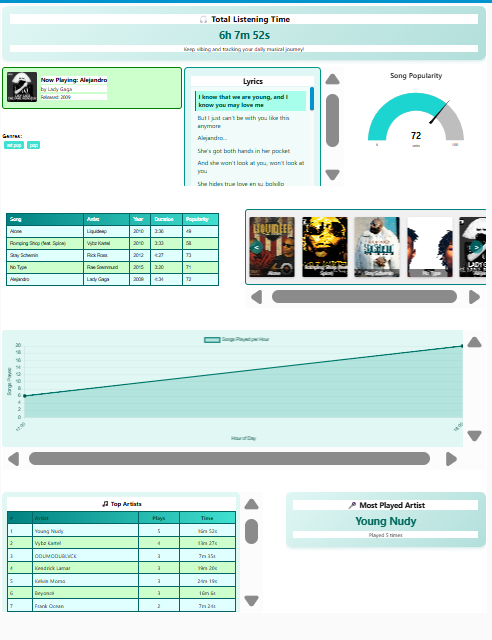
But what if you didn’t have to wait until the end of the year?

This project was born from the idea of making passive listeners into active listeners. With this real-time dashboard, users will be able to access their music analytics as they listen. Instead of relying on Spotify’s end of year summary users will be able to know about their top artists, song trends, genre preferences and listening time instantaneously.

When users are actively engaged with their listening habits, they are able to get a sense of their mood, routines, and identity allowing them to stay more connected to the music they love.

This dashboard was created using **Node-RED,** which is a flow-based development tool, and the Spotify API.

# SCREENSHOT OF FULL DASHBOARD VIEW



# **SYSTEM ARCHITECTURE AND FLOW OVERVIEW**

The dashboard is powered by **Node-RED,** a flow-based development tool ideal for connecting APIs processing data and building UIs with minimal code.

Full Node-RED Flow Diagram

A diagram of a company

AI-generated content may be incorrect.

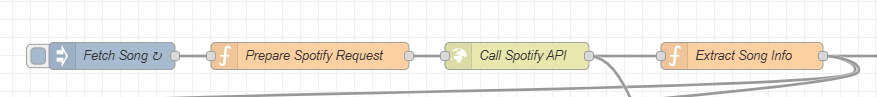
Authentication and Token Management

**A diagram with text on it

AI-generated content may be incorrect.**

These Nodes are responsible for establishing a secure connection with the Spotify API using OAuth 2.0 protocol. The user can get an authorization code which is extracted and used to authenticate all API requests. This connection allows for seamless connection to the Spotify service without needing repeated user intervention.

Fetching Current Song Data

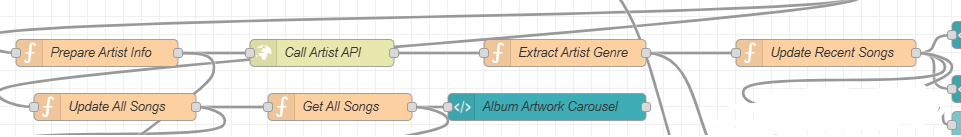
****

This flow is responsible for retrieving real-time information about the currently playing track from the user’s Spotify account. The Fetch song is automatically triggered after 5 minutes which initiates the process of song information from Spotify to be extracted. Key details like the song name, artist, album art, release year, track ID, duration, and popularity score are extracted. This data is then fanned out to multiple downstream flows including Now Playing Art Cover, Recent Songs Table, Popularity Gauge, Genre Definer, and lyrics-related flows.

This is the output that should be displayed in the debug Node.



Song Logging and Historical Tracking



This is at the core of building a continuous, real-time listening profile. The Update All Songs function node starts by listening to the song data that has been parsed from the Extracted Song Info node. The function of this node is to store each individual song that the user has played. Using its unique id, it determines whether the current music differs from the most recent entry. If it is new, it adds the song object to an array that is kept under the key allSongs in Node-RED's flow context. The majority of the dashboard's statistics are powered by this array, which expands with time to become the primary historical dataset. Other flows that extract and aggregate historical metrics from this collection are Get All Songs, Generate Timeline Data, and Generate Top Artists.

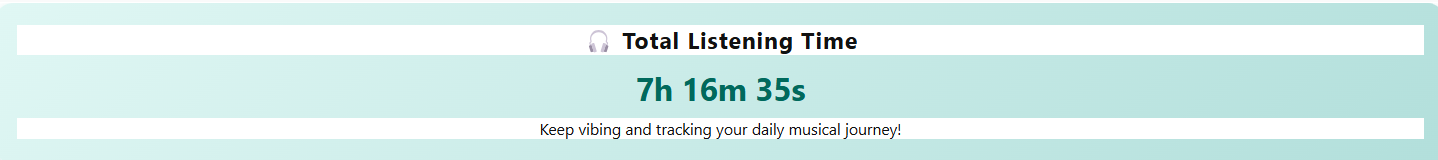
Analytics and Visualization

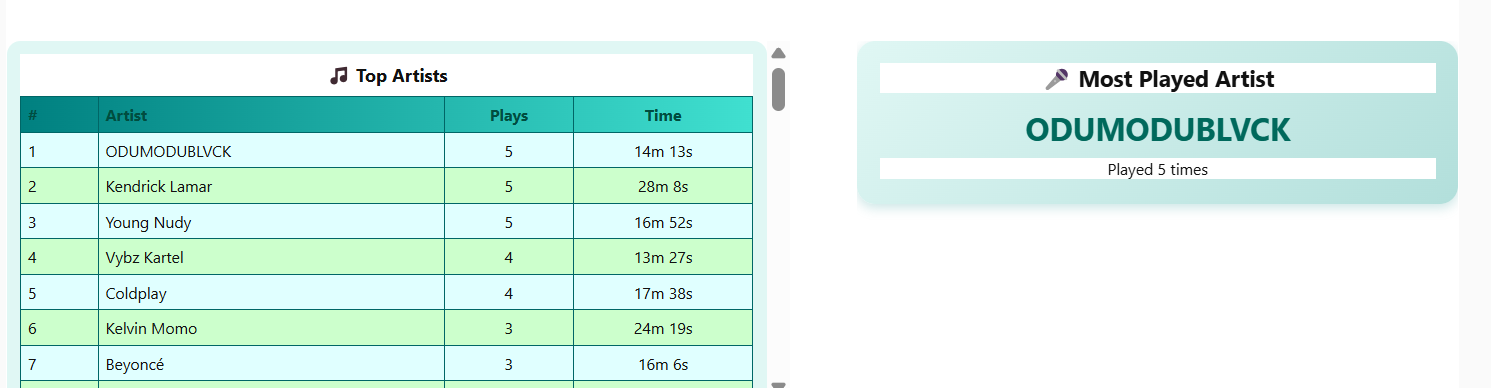
**A diagram of a music system

AI-generated content may be incorrect.**

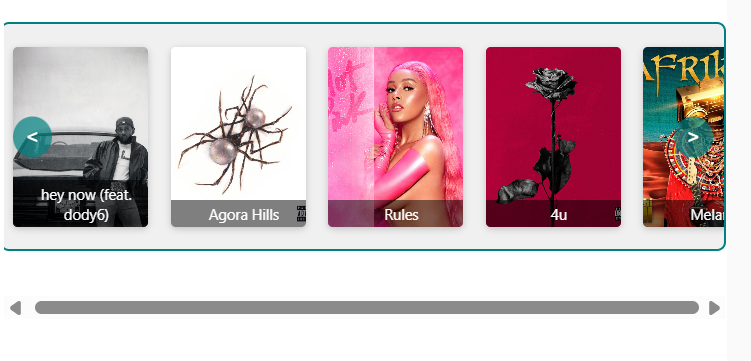
The nodes in blue are the visualization nodes. They convert the data into dynamic, interactive graphics that add interest and information to the dashboard. Using a variety of ui\_template, ui\_chart, and custom Chart.js elements, it extracts and aggregates insights to be displayed, primarily from the allSongs array. The number of songs played at each hour of the day is calculated by the Generate Timeline Data node using timestamps from each logged song. The Listening Timeline ui\_template node receives this data and uses Chart.js to create a dynamic line chart that helps users identify the times they listen to music the most. Similarly, the Generate Top Artists node iterates through the song history, counts the number of times each artist has been played, and optionally sums their listening durations. This is then visualized as a **ranked artist table.**

A function node is used in the **Total Listening Time** section to aggregate the duration (in milliseconds) of every song and transform it into a readable format, such as "1h 45m 30s." A stylized ui\_template card receives this value and displays a clear and appealing summary of the user's overall Spotify interaction. Similar to this, a **Most Played Artist Card** shows the name and total number of plays of the top artist inside a teal-highlighted summary card, dynamically identifying them based on play count. Together, these images—all included in a themed, scroll-free user interface layout—not only give a real-time summary of the user's actions but also promote thoughtful listening by providing details like mood, time patterns, favorite artists, and cumulative consumption.





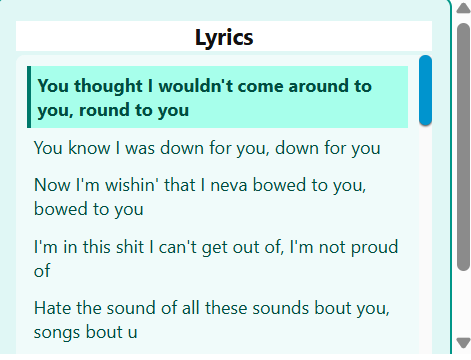
The **Album Artwork Carousel** and **Now Playing Card** components serve as the dashboard’s primary visual anchors — designed to make the user interface not just informative, but also enjoyable and aesthetic. The **Now Playing Card** is updated in real time using the output of the Extract Song Info node. It displays the currently playing song’s title, artist name, album art, and release year, all neatly wrapped in a bordered, styled card with a green background and rounded edges. This instantly shows the listener what’s playing without needing to switch to the Spotify app.

 A screenshot of a computer

AI-generated content may be incorrect.

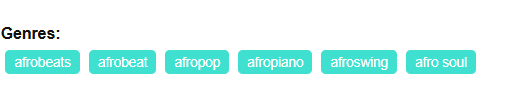
The **Album Artwork Carousel**, built using a custom ui\_template node, is a horizontally scrollable shelf-style slider that displays the cover art of all songs logged in the allSongs array — not just the recent five. Each album cover appears like a book on a shelf, complete with a title caption overlay at the bottom. Navigation arrows on either side allow users to scroll through their listening history visually. The carousel is programmed to always display the newest song at the far right and updates dynamically whenever a new track is added. The use of AngularJS within the ui\_template allows for interactive control, and the carousel aligns with the dashboard's teal/turquoise theme.

The **Live Lyrics Syncing** feature provides real-time display and highlighting of song lyrics alongside playback. This system begins with a query to the **Genius API** (or an alternative public lyrics API) using a cleaned version of the current song’s title and artist. Once the lyrics URL is retrieved via Genius Search and Extract Genius URL, a request is sent to retrieve the full lyrics content. The Format Lyrics function node splits the lyrics into individual lines and estimates how long each line should be shown by dividing the track’s duration (duration\_ms) by the number of lines. Using the Spotify progress\_ms, the system highlights the appropriate lyric line based on the current playback time.



A ui\_template card renders these lyrics in a vertically scrollable container styled with a teal background and white text. As the song plays, the lyrics auto-scroll, and the active line is highlighted for clarity. If no lyrics are found or if the playback is paused, the lyrics panel hides automatically. This flow adds a deeply immersive layer to the dashboard, allowing users to follow along with their favorite tracks in real time. The downside is that the lyrics function only works for a limited number of songs.

The **genre visualization** component enhances the dashboard by providing context about the musical styles users listen to most. Each time a song is played, the artist's metadata is retrieved via the Spotify API, which often includes a list of associated genres.



The **popularity visualization** provides users with a quick and intuitive representation of how well a song is received globally. Each track on Spotify has a popularity score ranging from 0 to 100, which reflects its total plays and recent traction. This value is captured in real-time from the Spotify API and visualized using a **half-circle gauge** on the dashboard.

A blue and grey scale with a black needle

AI-generated content may be incorrect.

The **Song Timeline Chart** is a key analytical feature designed to visualize the user's music listening habits across different hours of the day. Each time a song is played, the system records a timestamp and stores it in the allSongs array within Node-RED’s flow context. The timeline flow processes this historical data by grouping songs based on the hour they were played.

