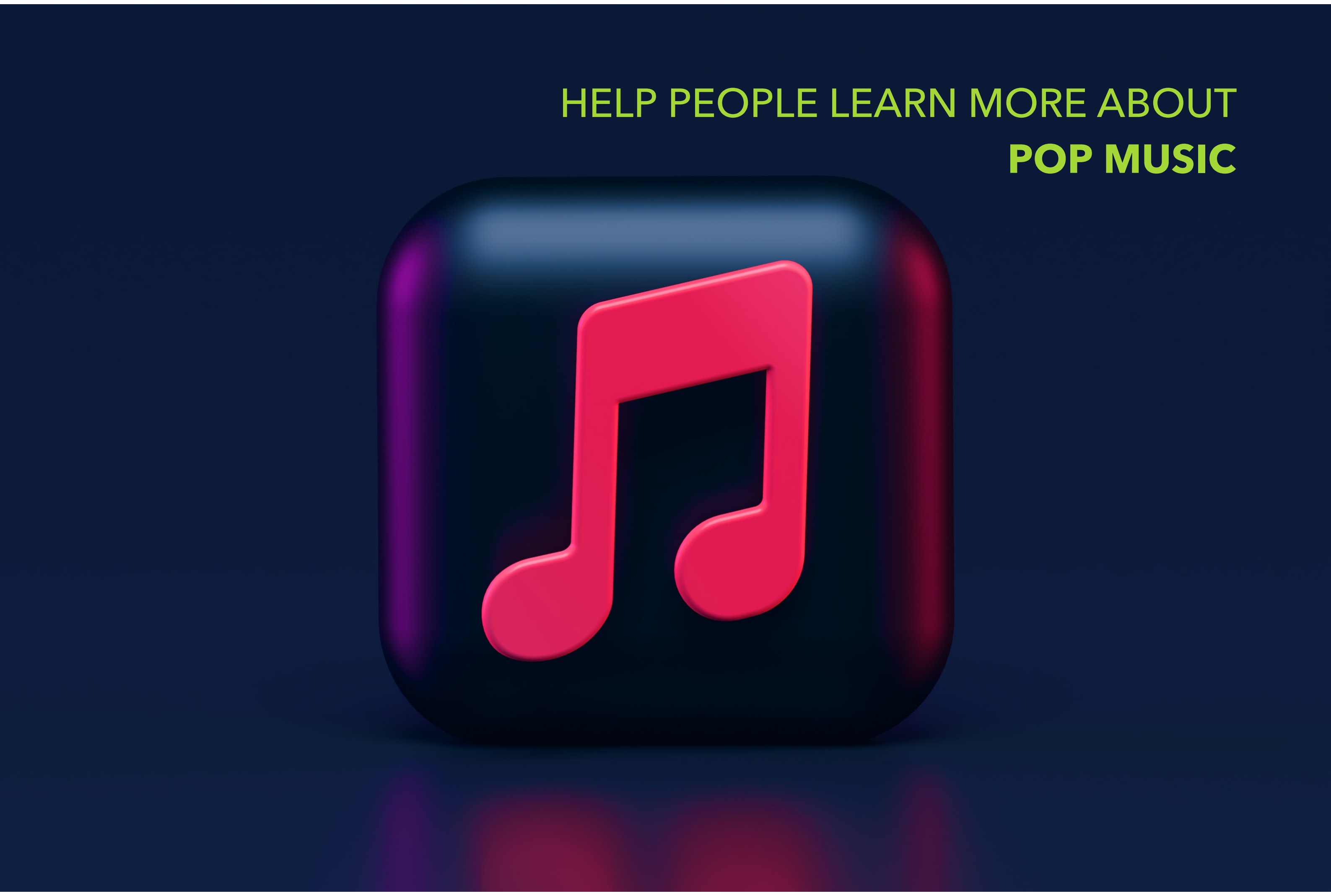


FINAL REPORT

DISCOVERY OF POP MUSIC

OUR MOTIVATION

Our initial idea came from the personal experience of one of our team members. His mother, a 53-year-old English teacher, had recently grown a love to electronic music by scrolling on TikTok and had developed an interest in popular music nowadays. So she asked her son about famous artists in pop music and their style and emotional expression of music.

Therefore, we realised that a significant portion of the population was new to pop music, and more and more people are turning listening to tune into a daily habit. Our objective is to help these individuals to find the music they appreciate.

OUR GOAL:

By introducing them **to the evolution of pop songs over four years** and **to the genres and cultures of famous pop artists**, we enable them to find their favourite artists and genres to facilitate their next musical exploration. In the table below, we have broken down the objectives even more into five specific goals. And based on the five objectives we have decided which data to look for.

GOAL1	GOAL2	GOAL3	GOAL4	GOAL5
The musical style and characteristics of each singer	The different moods and cultural expressions of the singer	The singer's musical style from period to period	The pop trends	The popularity and the common taste
DATA1	**DATA2**	**DATA3**	**DATA4**	**DATA5**
The general characteristics of all the singer's songs	The frequency statistics of the lyrics	The variations of the singer's musical characteristics	The variations in the characteristics of popular music	The songs which is long-standing in the top 100 music board

DATASET

After defining our objectives, the dataset that we looked into was from Kaggle, but it provided far too less information. And after the exploration, we found more information can be obtained from a combined usage of Spotify API, Genius API, and QQ music API, so we decided to **assemble our own dataset**.

The main data source that we used is **Spotify API**, **Genius API**, and **QQ Music API**. For the purpose of our visualisation, we need data of some particular singers and the top 100 tracks from the past years. In detail, we utilise the data source in the following ways:

- From **Spotify API**, we gather all albums and track data related to a singer, along with audio features for each track.
- From **Genius API**, we fetch the lyrics of tracks that we obtained on Spotify API, as Spotify doesn't have lyrics retrieval API endpoint.

- From **QQ music API**, we gather data on billboard weekly top 100 tracks and weekly top 100 trending tracks in the Chinese market since 2018. The lyrics are available with each track with the QQ music API, but no audio features are available, so we query each track's audio features from the Spotify API.

All the fetched data are stored in the Github repo in JSON file format. The API fetching code for Spotify and QQ Music are listed inlined.

Most of the data returned by all APIs is relatively clean regarding the preprocessing. The only part that requires some preprocessing is the lyrics coming from Genius API and QQ music API. Since both of them are returned with timestamps (pointing to the time when the lyrics are sung) and/or some miscellaneous information (e.g. singer, album company, etc.), we will need to strip them out before analysing the lyrics.

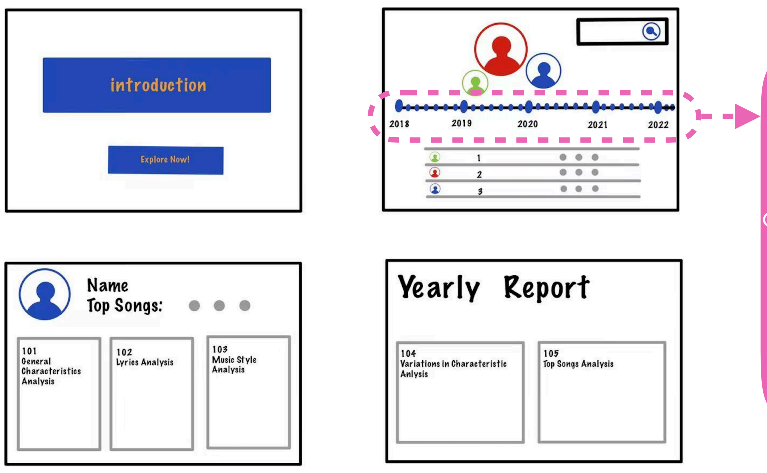
LIBRARY SELECTIONS

Since our group members are new to the front-end, we unanimously decided to use the relatively easy-to-use tools. As for the JavaScript Framework, we chose the **Vue.js** because of its small size, good encapsulation and ease of use. As for the visualisation library, we decided to use the **Highcharts.js** due to its excellent and convenient interaction methods and a large number of examples.

The data in use is stored on Github course project repo, using Vue.js to fetch the raw files from it, and then parse the data and process it to feed it into the Highcharts.js visualisation. The data format being stored in the repo is always JSON, which is friendly to be operated on.

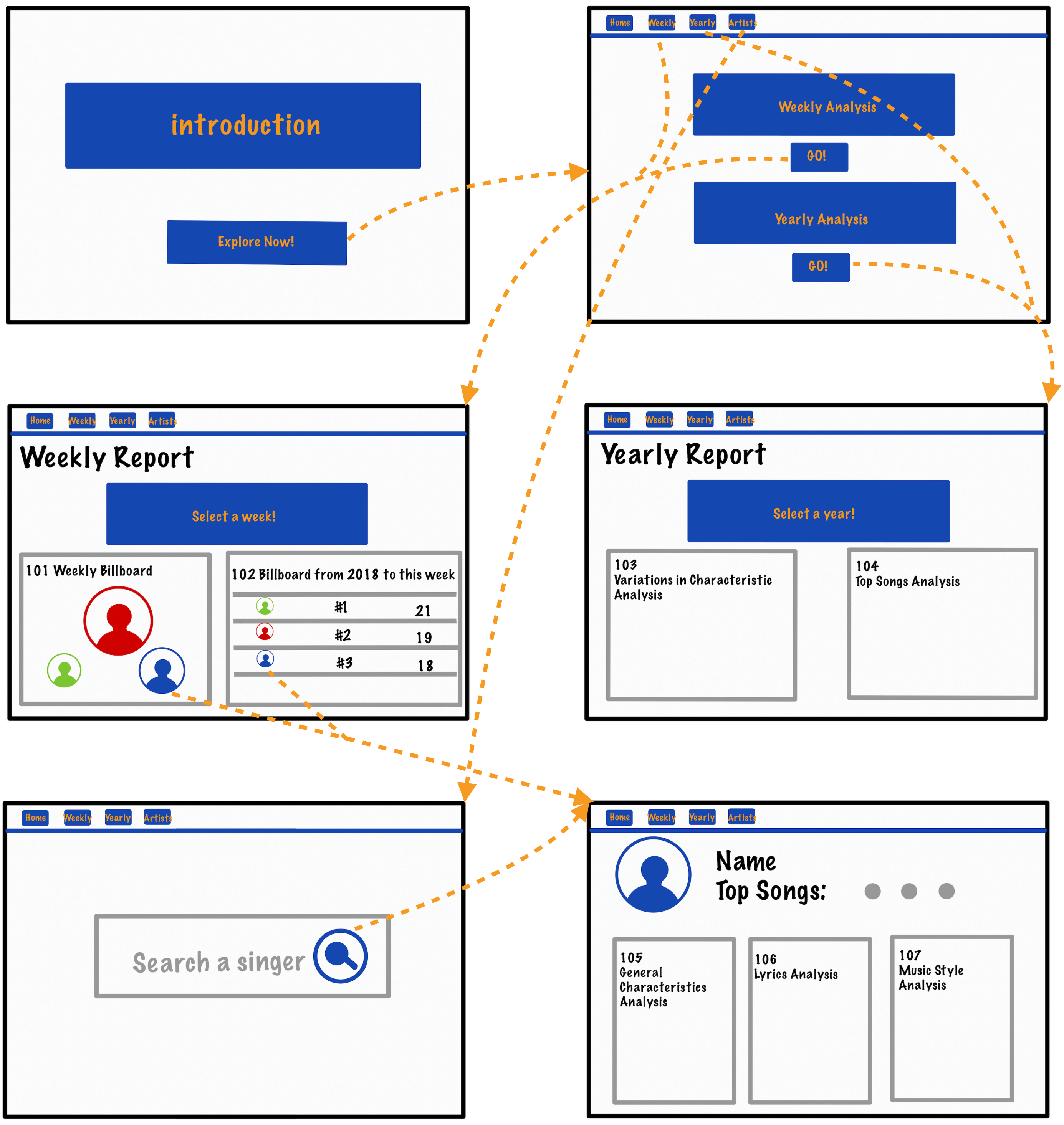
SKELETON OF WEBSITE

In Milestone 2, we integrated the five datasets identified in Milestone 1 into the skeleton of our website. However, in the website implementation, we found one main problem.



The problem with this is that if the options for weekly analysis data and annual analysis data from 2018 to date were to be integrated on a timeline, the visualisation would be unclear and would lead to poor user interaction with the data.

To solve this problem, we have set up a dedicated page for users to choose whether to view weekly analysis data or yearly data and separated the search. Additionally, in order to enable users to find the corresponding visualisation more quickly, we have set up a side bar at the top for quick selection. The final skeleton of our website turned out to be the image below.

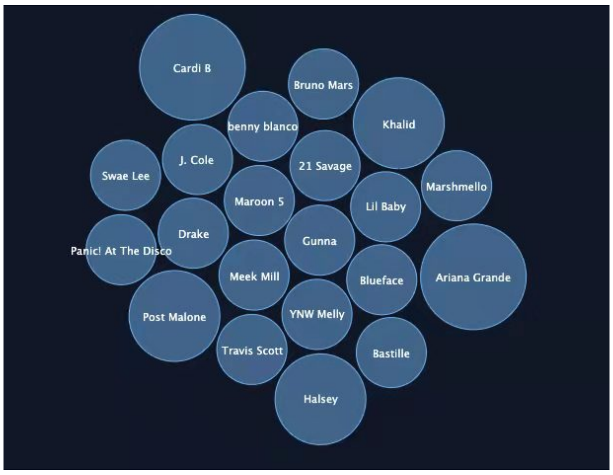


As a result of the changes, the content of each screen has become more independent and the visualisation and interaction of each section has become cleaner and clearer.

VISUALISATIONS

We offer three different ways to explore the world of popular music: trending of the week, hits of the year and finally, the profile of singers. Overall, we have **11** charts, of which there are **6** types in total. We will then talk about why we chose these five visualisations to represent our data and how they correspond to the skeleton of the website.

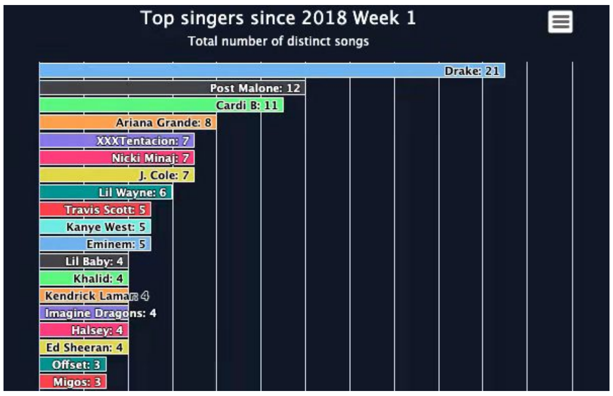
Bubble chart - 101



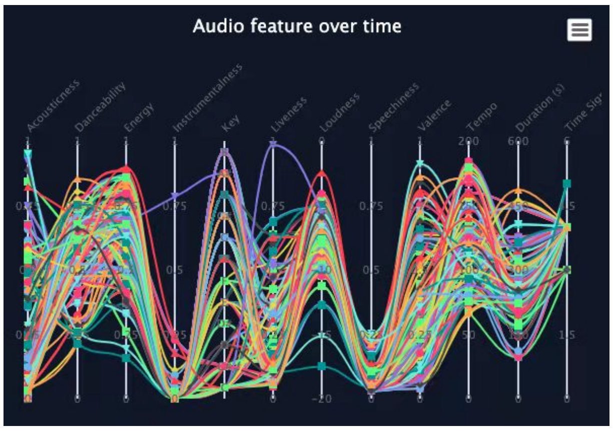
Who are the most popular artists that have the most listened to songs from one week to another? It is possible to have this information directly by contemplating our bubble chart. This chart represents the singers, whose songs figured in the top 20 of the given week. The larger the size of the bubble is, the more popular his songs are this week.

Histogram - 102, 104

This histogram shows the ranking of artists with the most significant number of accumulated songs on Billboard from the beginning of 2018 to the given week in a weekly report. We also use the histogram to show the top singers sorted by the number of songs on Billboard and the duration of the songs staying on the billboard in the yearly report.



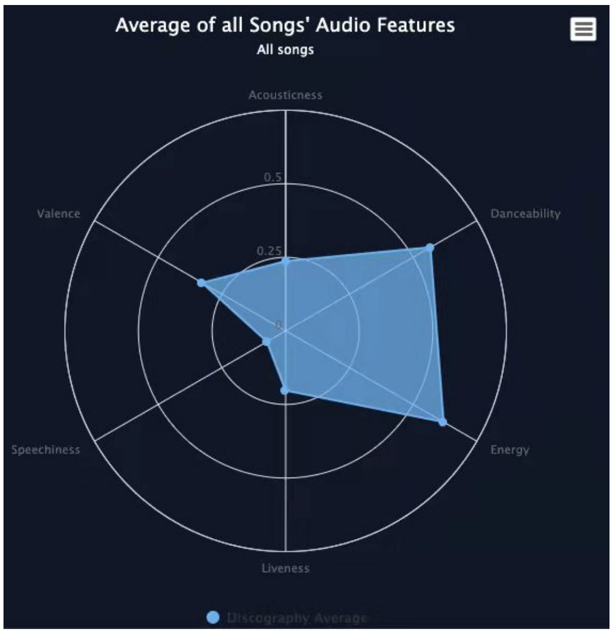
Parallel coordinates - 103



We use the parallel coordinate to show the audio features of the year's top singles to showcase the trends of this year's top singles. The main reason we use parallel coordinate is that the number of hit singles is over 100, and using a line to represent a song will aggregate all the information more clearly.

Radar chart - 105

We noticed that there are lots of interesting audio features that come along with each song in the dataset of Spotify, which characterise the song. We also observed that among them, many take a value between 0 and 1. It came to us to draw a radar chart for each song to visualise its characteristics. A radar chart is suitable to display multiple continuous variables that are on the same scale, and these variables do not necessarily have a strong link between them. The goal is to have a global view of the variables and to compare multiple items on these characteristics.



- **Acousticness** shows a confidence measure of whether the track is acoustic.

- **Danceability** describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat, strength, and overall regularity.

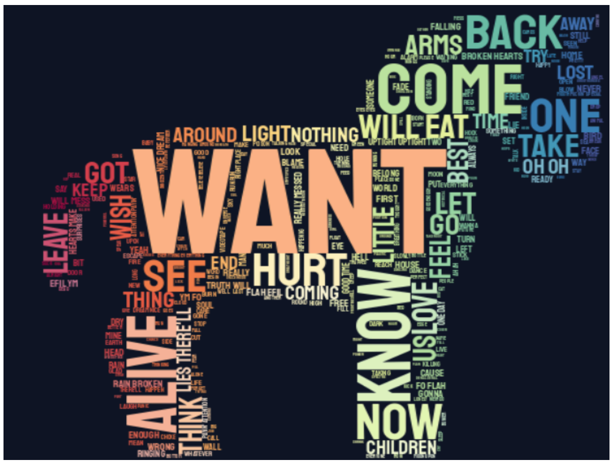
- **Energy** represents a perceptual measure of intensity and activity; it takes into account dynamic range, perceived loudness, timbre, onset rate, and general entropy.

- **Liveness** detects the presence of an audience in the recording.

- **Valence** describes the musical positiveness conveyed by a track; the song is more cheerful when valence is close to 1, sad or angry when it is close to 0.

- **Speechiness** detects the presence of spoken words in a track.

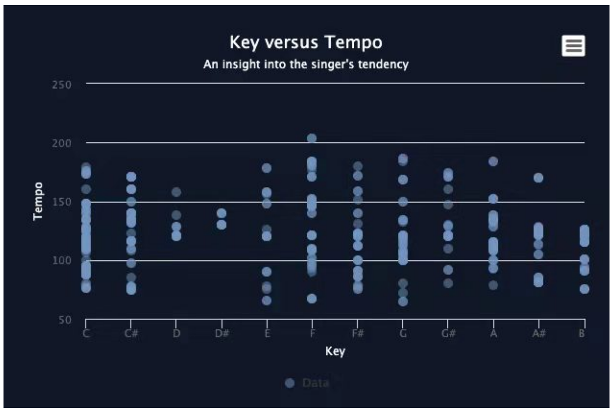
Word Cloud - 106



We also perform lyric analysis using a word cloud, which provides a clear insight into the most frequently occurring words for a given singer or a given year. The larger the font size of a word is, the more times it appears in the lyrics; it highlights the popular words of the given context. We used a stop words dictionary to filter out the meaningless words.

Scatter plot - 107

To complete our exploration of audio features, we visualised the tempo and the key of all songs for a given singer. We utilised a scatter plot with tempo on the x-axis and key on the y-axis. Since scatter plots are used to show relationships between variables, this can give us a sense of the artist's favourite keys and tempos and their relation. When we observe a cluster on the plot, it suggests the artist loved to use a particular key and tempo combination.



IMPLEMENTATION DIFFICULTIES AND CHALLENGES

We have around **200 singers' data** and **several hundred tracks of audio features**; we need to load and extract data instead of having everything preloaded dynamically. We eventually used **Vue.js + Nuxt.js** to build our website because we have a lot of interactions, and Vue.js is excellent for reactive work, and structuring them using components in Vue.js is easier to manage. We hosted our dataset on the Github repo in the format of JSON. We then use the async Javascript function to fetch the raw JSON file and perform the necessary computation.

We ran into issues mainly in the Nuxt.js framework, especially the integration with HighCharts.js. **The plugin system from Nuxt.js is not very friendly with HighCharts.js**, especially when using special graphs, such as the parallel coordinate graph. The importing error was too hard to debug, so we eventually fallback to using <script> to load the minified JS and CSS files over CDN during runtime. Also, updating the graph dynamically using the Nuxt.js plugin system is challenging.

Properly structuring CSS to **style the webpage** is also a big challenge. But TailWindCSS comes to the rescue. It's much easier to write CSS code that works; as we decided to go with dark mode, everything needs to be styled. TailWindCSS has one minor drawback: compilation is required, and although it makes life easier, the nitty-gritty of properly adding

CSS styling code at the right place is still non-trivial and complex to get done right in the first place. For example, to make the background fully dark, from top to bottom, it took us quite some time to fix the white bar at the bottom of the screen.

Making the website aesthetic is sometimes tricky, especially when we are doing it in **dark mode**. We have to reverse the background colour of some graphs and pick the appropriate colour palette that is neither too flashy nor too dull.

Due to the **slowness and rating limiting of the Genius API**, getting the lyrics data is very time-consuming. Compared to other musical data visualization works, we chose to include more artists and use three different datasets. This generates a lot more data, which consists of a challenge for the **website's capacity and slows down the processing time**.

Another challenge is the **incompatibility of website displays for desktops and smartphones**. We only designed the website layout using CSS for the desktop version, not the mobile version. The display on phones may differ from the effect we desire because the web page structure needs to be reorganized for the mobile layout to fit the different screen sizes.

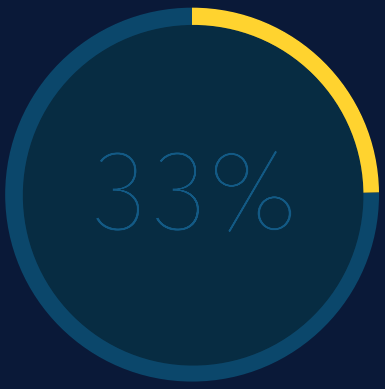
Since we are processing data on the fly for all the graphs (200+ singers, 200+ weeks, we dynamically generate 3~4 feature graphs per page), sometimes this will reach the performance **bottleneck of the browser**. Currently, speed up is done using placeholder and async data loading techniques, so the user will gradually see the page "hydrated".

Presently, tuning is done to speed up loading using some placeholder, but spinners might be future work. Nuxt.js + Vue.js is very friendly to be paired with TailWindCSS, but the plugin system is painfully hard to use, especially when integrating with the 3rd party plugins for d3.js and highcharts.js. In the end, we fall back to using good-old CDN, by adding it using the <script> tag, in the HTML <body>.

PEER ASSESSMENT

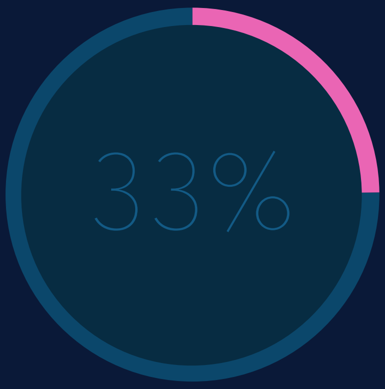
Every member had to do some data processing on python for some features. For the feature implementations and general workload, the separation was as following.

Everything is reviewed by other team members and choices are made together. The main workload for a feature is usually done by one person, with other members also verifying and modifying as needed.



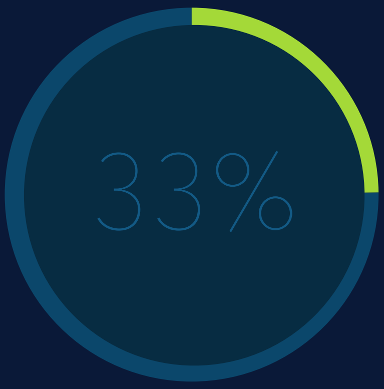
CHUN-HUNG TSENG

the build of the website framework; and 5 visualisations



ZIMI LIU

3 visualisations; process book



YINGXUAN DUAN

3 visualisations; process book