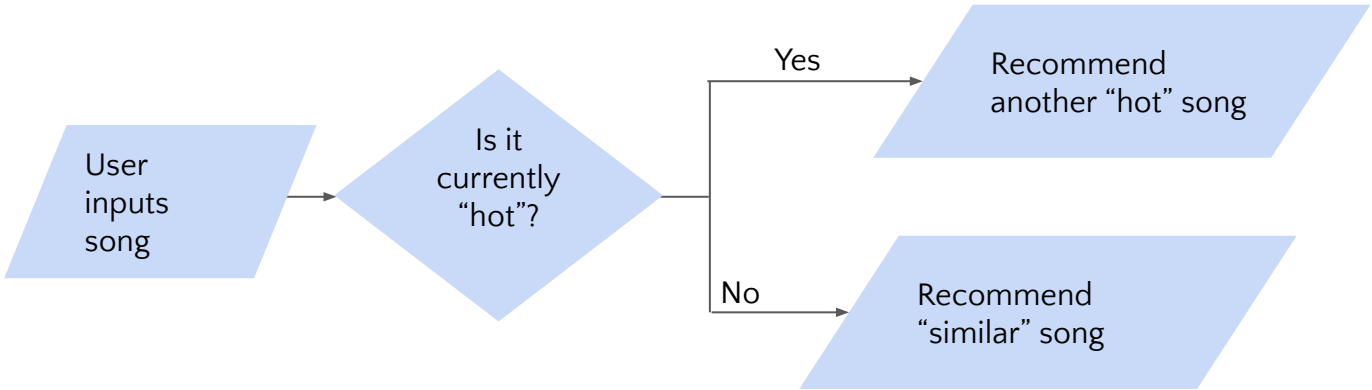
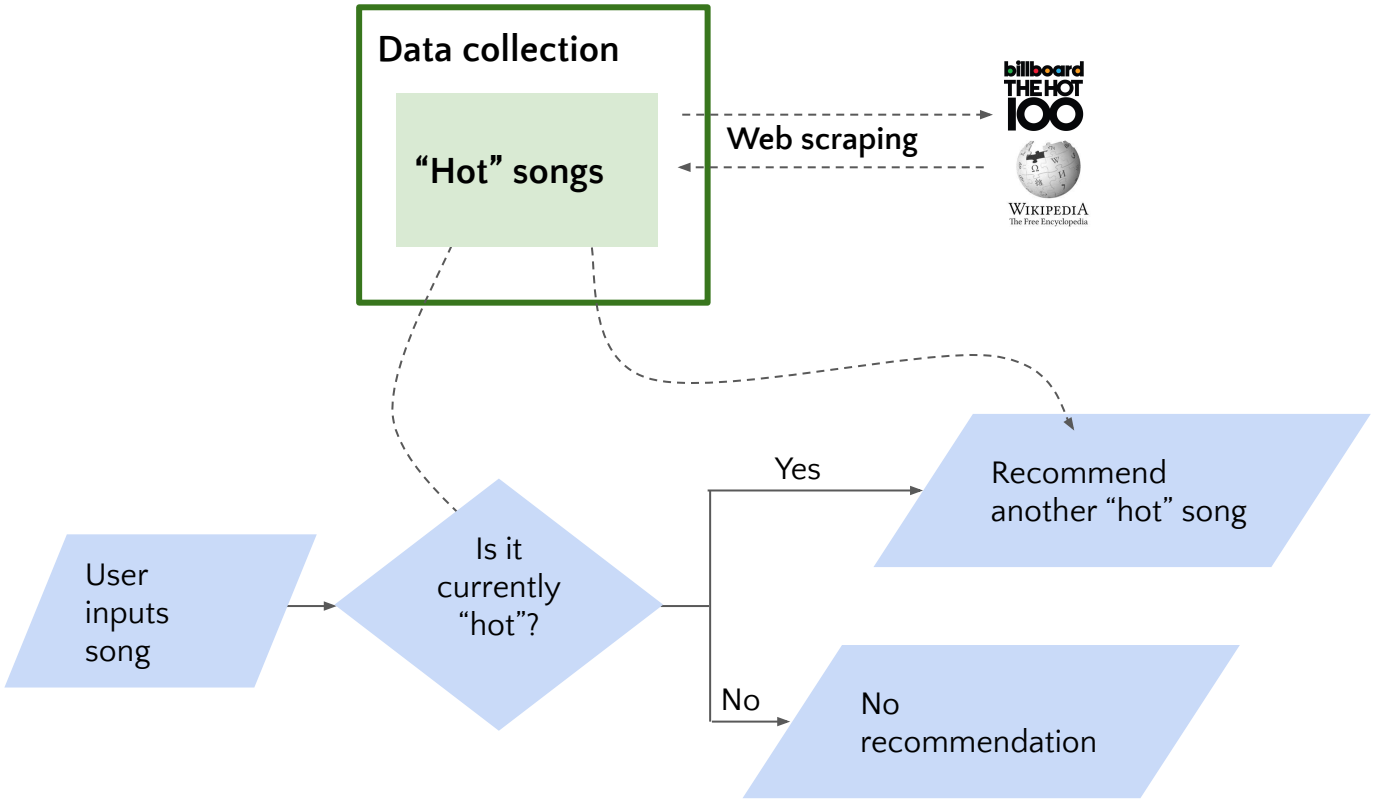
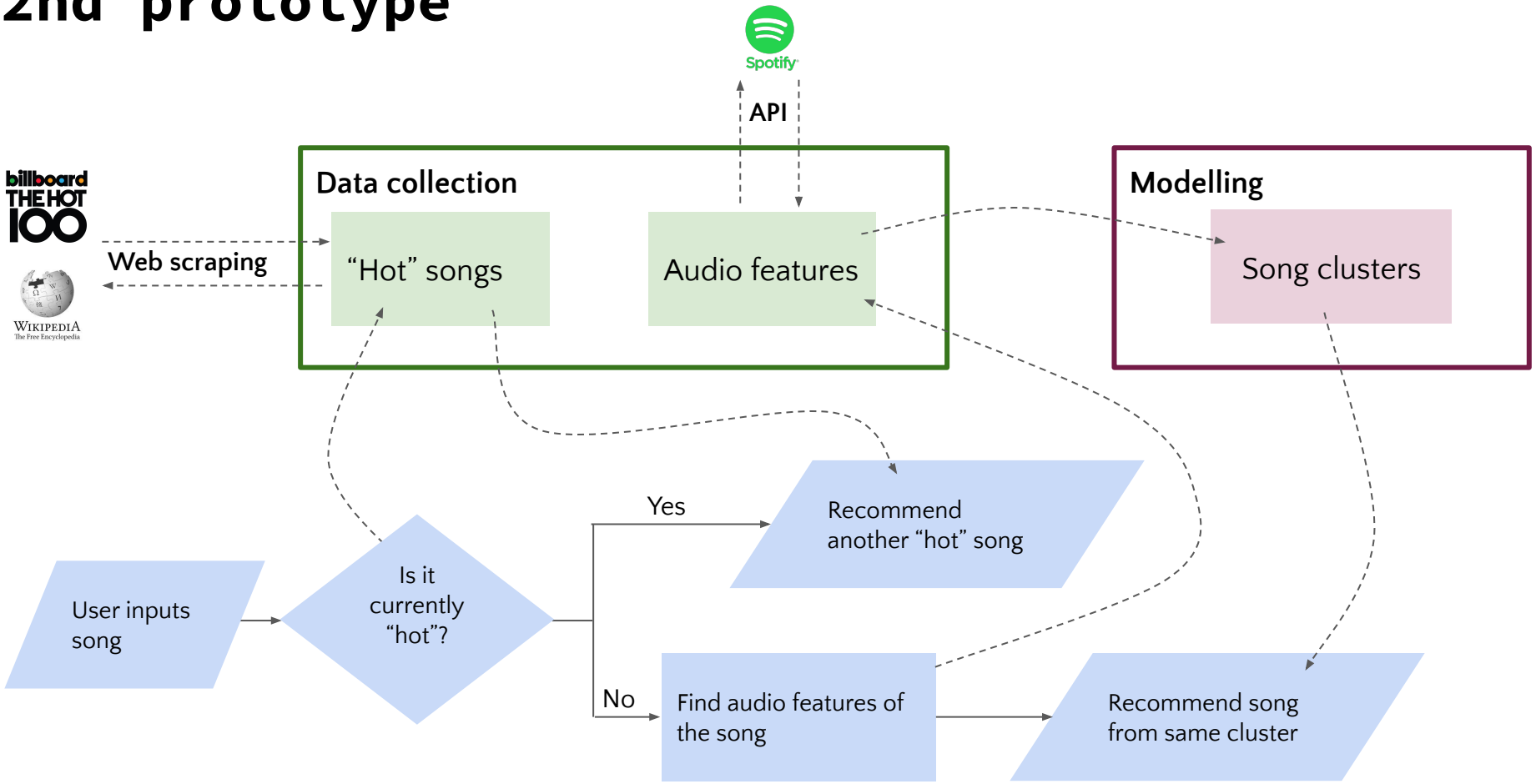
Project flowchart

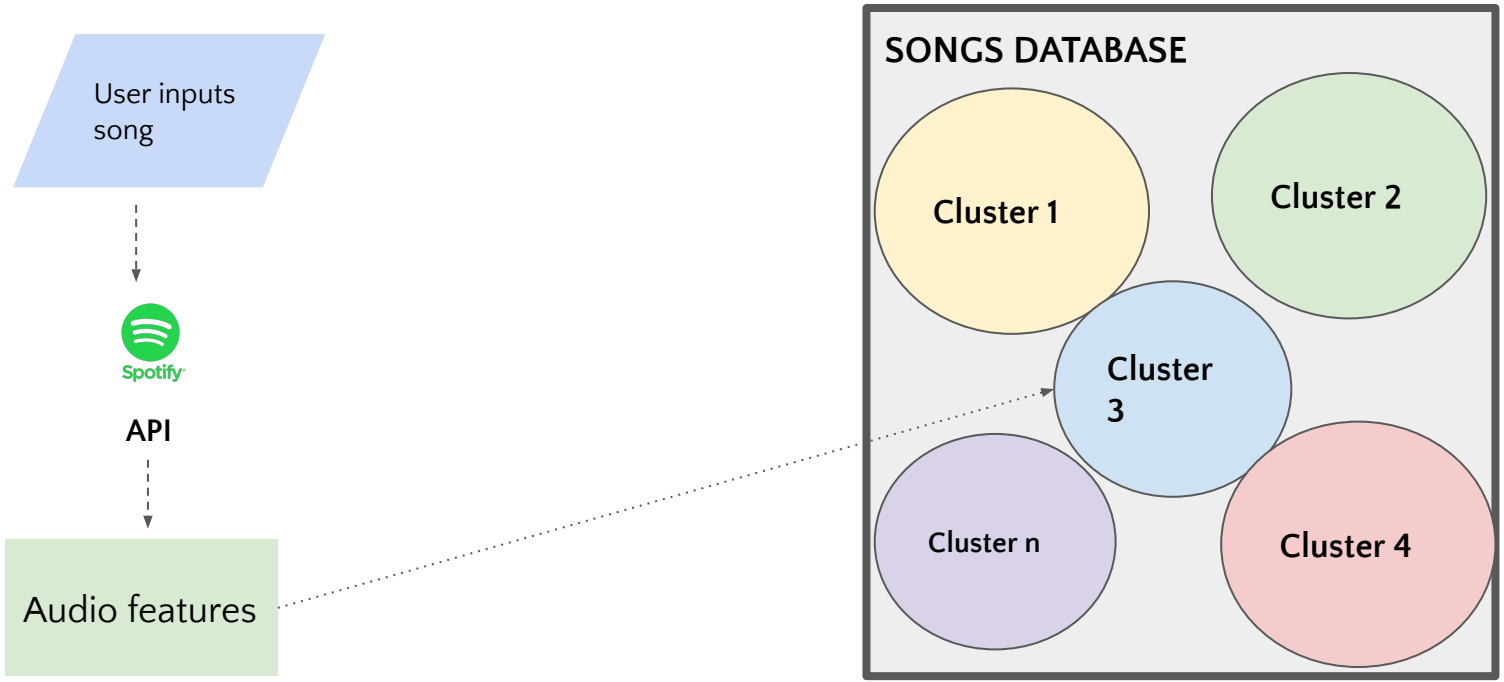


1st prototype



2nd prototype




User inputs song
Spotify
API
Audio features
SONGS DATABASE
Cluster 1
Cluster 2
Cluster 3
Cluster n
Cluster 4

In this final part of the project, you should be focusing in 2 big areas:

1. **Cluster the songs you collected:**

- scale the audio features of your songs. this should create an object called `scaler` (store it, you're gonna need it in the future) and an array with scaled features, let's call it `X_scaled`
- initialize a KMeans model with `kmeans = KMeans(random_state=1234)` (don't waste time on parameters /number of clusters for now - use defaults!)
- fit the model to your data using `kmeans.fit(X_scaled)`
- create a column called cluster in your original dataframe, with the assigned cluster, using `X["cluster"] = kmeans.predict(input_song)`
- this process should only be done once, not every time a song is inputed! However, you are going to need the clustered dataframe `X` , the `scaler`, and the `kmeans` model to be loaded in your environment (i.e. notebook) when the user inputs a song. Tip: consider doing this through creating a module and loading it from another notebook.

2. **Assemble the project pipeline:**

When the user inputs a song, you should be able to:

- receive an input song from a user. let's imagine it's *Bohemian Rhapsody*
- check if *Bohemian Rhapsody* is hot. it's not, so…
- send "*Bohemian Rhapsody*" to the Spotify API and get its audio features. store them in a variable called, for example, `song_audio_features`
- scale the audio features using `song_scaled = scaler.transform(song_audio_features)` (this is the `scaler` we created above!)
- get the cluster of the song, using `kmeans.predict(song_scaled)` (this is the `kmeans` model we created above!). Let's imagine it's cluster 3.
- rom your dataframe of collected songs `X` , get a song that belongs to cluster 3. Let's imagine it's *Stairway to Heaven*.
- print *Stairway to Heaven*: this is your recommendation!