

# Song Search & Play App: Pseudo Code

Illustration #1 (Dekstop/Tablet configuration)

Note: We will probably want to reverse the postion of Divs 7 and 9 (bottom right) from what is shown in this picture. The reason being that if we put the lyrics at the bottom, we can allow users to scroll down in desktop and mobile without breaking experience.

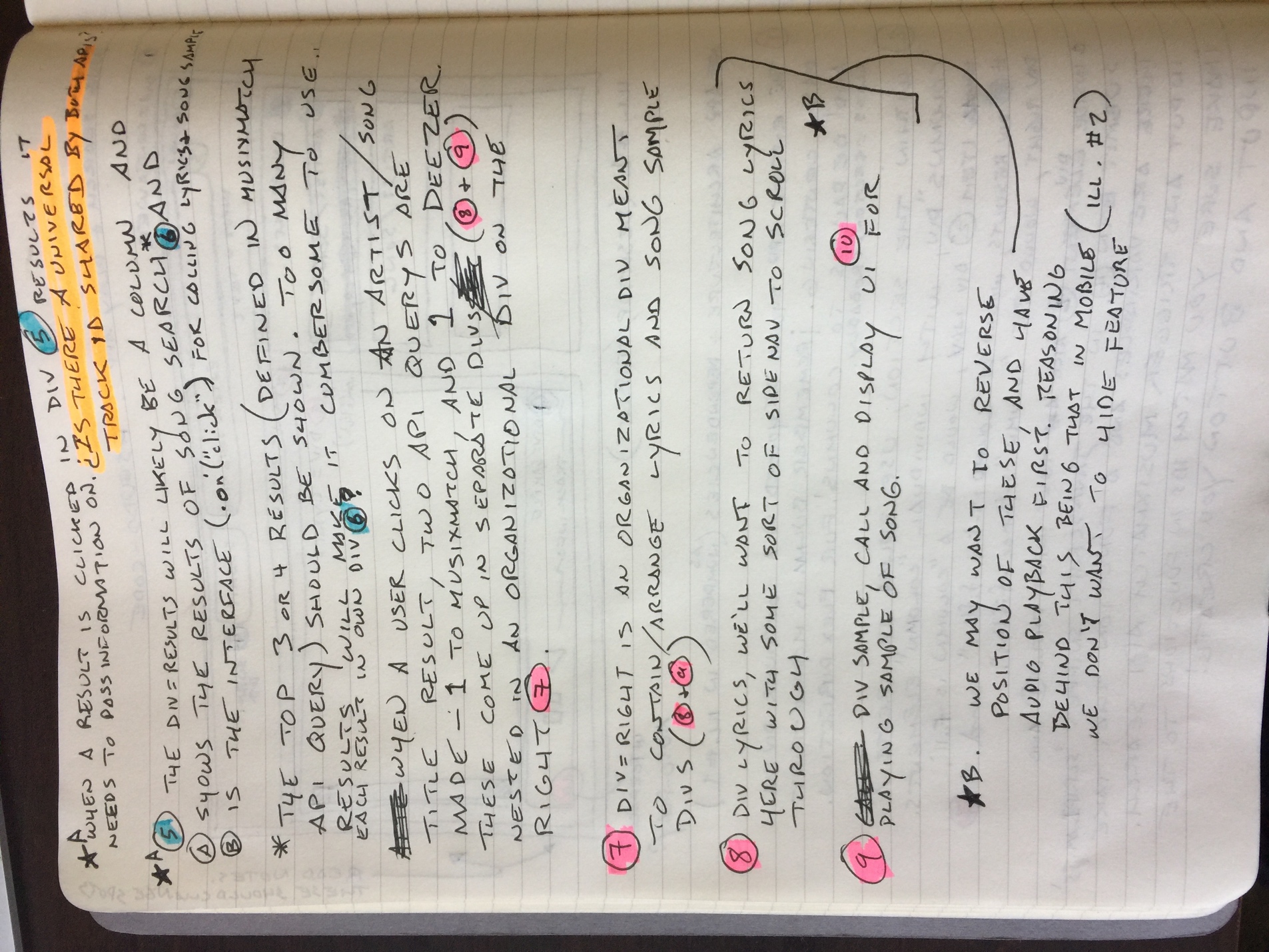
**App Architecture and Dependencies**

**(1)** The entire app should be housed in a container Div for easy centering. **Remember that Bulma is Mobile First and defaults to Columns for flex direction.** Class should be set to .section probably.

**(2)** Within the section, use Bulma to create a “Columns” div, with individual “column” elements inside. **For example,** Item **(2)** Div Nav, would be a “column is-full”. Item **(5)** Div Results, would be a “column is-two-fifths”, and **(7)** Div Right, would be a “column is-three-fifths”.

Inside of Div Element **(2)** are a text input field **(3)** for entering the song title and a submit button **(4).** In the javascript file “script\_AM.js” there are declared variables and a function written that take input from a text box and trigger musixmatch API searches on that term.

* Make sure that you match IDs called in the function to the IDs you assign to the input and button that you create!
* Also, the musixmatch API is currently returning misformatted URLs to view the song lyrics. It seems to be inserting a backwards slash in front of every forwards slash (e.g. \/ ), which is causing the resulting URLs to be broken. Let’s discuss with Iryna and see if this is a result of the additional CORS elements we added to queryURL.



**Comment A:** When a search result is clicked in **(5)** Div Results, it then needs to pass information on to two separate API querys. **Is there a universal Track ID value that is utilized by both apps> If so, we can just pass this, which will be much easier. If not, we need to constrain search term.**

**(5)** The Div Results will likely be a “column” and should A) show the results of the submitted song search\* and B) is also the user interface (on click) for calling lyrics and song samples in **(7)**

Div Right.

\* We should only call and display the top 3 (or 4) results, which can easily be defined in the musixmatch API query. If we show too many results it will make the app cumbersome to use, particularly in mobile view. We may want to consider assigning each Artist/Song Title result to its own div element **(6)** for easier styling, control, automation.

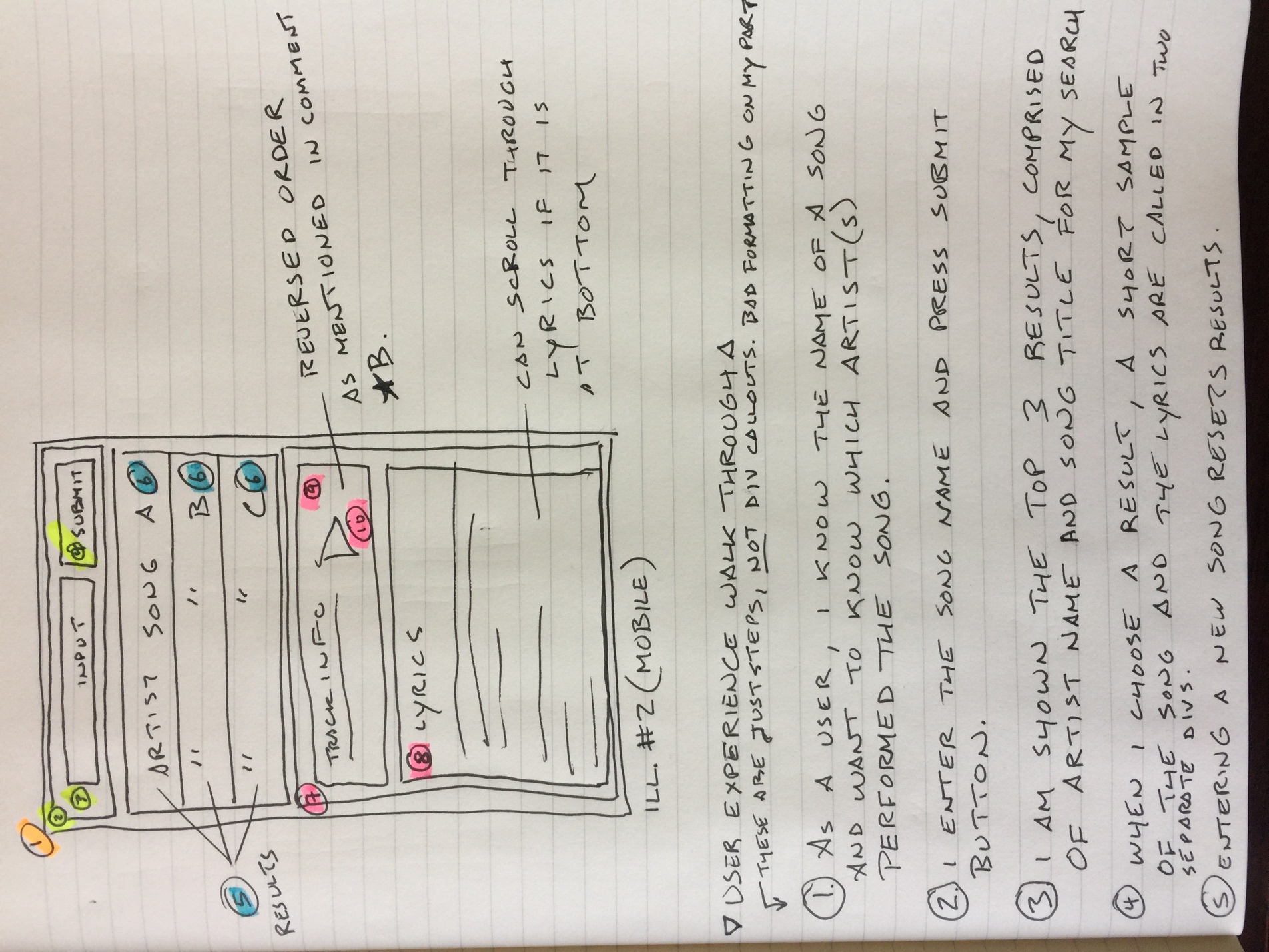
When a user clicks on an artist and song title result, two API querys are made – 1 to musixmatch (for lyrics) and 1 to Deezer (for song sample and UI?). These each come up in their own separate divs **(8)** for lyrics and **(9)** for song sample: these are both contained in the organizational **(7)** Div Right.

**(7)** Div Right is an organizational div meant to contain/arrange the lyrics and song sample divs **(8)** and **(9).**

**(8)** Div Lyrics. We’ll want to return song lyrics here with some sort of sidenav to scroll through them. **Alternatively, we swap places with (9) and put lyrics at bottom so they can down past fold without breaking user experience. This is probably a better idea.**

**(9)** Div Sample. Call and display UI from Deezer for playing a sample of the song.

**Comment B:** We may want to reverse the position of **(8)** and **(9)** , and have audio playback come first. The reasoning being that in the mobile configuration (Illustration #2), we don’t want to hide that feature.



**Note:** In Illustration 2, I have reversed the position of the lyrics (8) and sample (9) divs, to show what I believe is the preferable layout of lyrics at the bottom. This allows the user to scroll through the lyrics while the song sample plays, rather than having to scroll through all the lyrics just to get to that function.

**User Experience Walkthrough**

1. As a user, I know the name of a song and I want to know which artist(s) performed that song.
2. I enter the song name in the window at the top of the app, and press the submit button.
3. An API query to musixmatch, searching on the parameter “q\_track” is made, using the song title I entered.
4. I am shown the top 3 results of this search, which are comprised of the artist and the song title.
5. If I press any of these results, two new API calls are made. 1 to musixmatch for the lyrics, and 1 to Deezer for a short sample of the song. If possible, these are done using one universal track ID to simplify.
6. The results of these API calls are displayed in two separate divs, one for the lyrics, and one for the sample and any associated UI required to play it.
7. Pressing on the play button will trigger playback of the song sample.
8. I will be able to scroll through the lyrics.
9. Entering a new song into the search window, refreshes data, and generates new results.