After Action Report - SoundByte

Background:

1. Team Name: SoundByte (SSE Group 4)
2. Project Reviewed: SoundByte
3. Date of Review: 06/04/2021
4. When Review was Completed: During Project
5. Participants

Name	SID	Role(s)
Jiwoun Kim	200329205	Frontend design & Developer
Mason Lane	200376573	Scrum Master & Developer
Brandon Clarke	200373287	Backend design & developer

6. Project Summary:

In the year 2021 you're likely no stranger to being recommended music by some sort of software application, be it through social media or a digital media streaming platform. SoundByte aims to recommend songs for the sake of music creation; it does this by leveraging musical features that SoundByte extracts from the user's library. Many popular services that recommend songs to its users typically use very subjective features. SoundByte challenges the norm by using objective features to fully place the power in the hands of the creator. SoundByte's use of musical features relinquishes creative control to the artist while providing auxiliary guidance

In Software Systems Engineering one of the fundamental practices is to determine the context and scope of proposed software before any work begins on the software design. With that being said our app is given its context and scope through its users. These users are DJs, musicians, content creators, hobbyists and even the average music enthusiast. SoundByte aims to organize a user's music library in a fresh light and provoke new ideas on how the user's music can be used through suggested songs that would make for a good music sample, mix, mashup, or fade out.

7. Successes

Successes	Ensuring Success in the Future
The application returns song suggestions	Always return some sort of suggestion as that is one of the core requirements. Strive to continuously evaluate the quality of these suggestions and form a roadmap for improvement. Add further catches for the exceptions generated by the python script
Feature extraction successfully leverages Essentia in a web-worker & produces reasonably accurate estimates.	Further tests should be added to ensure the margin of error is truly reliable We should leverage our web-workers to follow best practices synchronously.
Filter function on suggestion results is intuitive and works better than expected	It gives users various options to find one for their preferred results. Add search functionality to extend the possible filters. Add filters to omit certain values.
Intuitive usability	When given the context of the purpose of our application it is very intuitive to use. Starting with providing the users song library sets the stage for the rest of the actions in the application. Functionality is neatly divided into tabs, this allows the different types of suggestion to be given context by one another. FInally every button that is associated with a function has an obvious purpose and is very user friendly in its language and physical placement. We should continue this design philosophy while introducing more synchronous actions. We aim for a low floor medium ceiling application
Playback function works with the DOM without having troubles.	It works correctly: when another play button is pushed,regardless of if they are playing or not stops them and plays the most recently played (pushed) music only.

Middleware/backend architecture has enabled a surprising amount of extendability with regards to future plans	We ought to leverage our current architecture to increase the types of allowable inputs, and the forms of suggestions our system is capable of. The middleware & backend might also help ease the load taken on by the frontend. Currently, our frontend handles filter logic. Middleware might be a better place to do this.

8. Need for Improvement

What Might be Improved	Recommendations
Playback design would be better if it includes progress bar to present current play position of the song file	Implement said features using html music players. Likely easy to implement after refactor
Feature extraction is a long process which disables interaction with the DOM.	Introduce synchronous feature extraction - displaying available data as it comes via the song menu. Fully extracted songs may be sent for suggestion. This will also allow us to support playback even when a song has not fully extracted features.
Encountered WASM memory issues during feature extraction given 200+ songs.	This problem may be resolved after introducing our synchronous solution, since the process is no longer all at once.
Each suggested result might be better if it has a unique id to save the results in a consistent manner.	It helps users to keep track of suggestions the user executed. This would also assist in our file naming conventions for saved results.
Measure of song similarity	More research could be conducted on the musical theory behind mixing music/when music sounds good. Currently it is a rather objective measure of song features and the differences between them.
SongMenu refactor would greatly improve the readability & extendability of our code.	Refactoring & writing more in Typescript would make our application more object-oriented. Refactor ought extract functions to create generic solutions that might help us reduce the amount of duplicate code.

	Refactors should mind the open-closed principle. Refactor should clearly define the difference between song lists and result lists in its code (don't do everything in the same file)
No automated testing or plans are present	Our refactor on the frontend would open us up to further options for automated tests. We should consider a proper plan. Currently, testing has been done - though in no formal capacity. We must be able to document our findings more accurately.
Few elegant instances of exception handling	Our aforementioned feature extraction improvements may open up new avenues of exception handling & ways to inform the user. See our code review for more on this.