-= 1 =-

The biggest extension that I completed is the browser-based musical player. This extension serves as the base for the visualization component. The main goal of this extension is to allow a user the ability to work with musical files, create his playlist, and then listen to the songs and visualize them, using the functionality of the template. I structured the player as the set of HTML elements that allow for interactive functionality. The code for this extension consists of three main parts. All HTML elements are just blocks that are grouped to create the user interface. To make them look visually pleasing, I add the style file, which consists of CSS code that makes our HTML elements distinguishable and adds some illusion of interactivity. But to add real interactivity, I use javascript code, that handles all these interactions. The first part of the code is the initialization of the variables, then, I preload the music component and then set up the visualization component. Code after this is separated into groups related to each element, like the "Play button" or "playlist", or some functionality, like "Show playlist" or "Load player settings".

The second extension is the timeline. This one is closely related to the player elements but is a bit tricky. It works both as the "user input element" and the "user visual interface". While the song is playing it shows current time and progress. But if a user clicks this component then it handles the navigation. All these interactions with the user are asynchronous in time, so I used built-in functionality to create "listeners" that wait for the trigger event and execute the code. This extension is just the part of the player, so it fits in naturally. The structure of this extension is also pretty simple. It's just a bunch of helper functions and one handler that uses them in the appropriate situation.

The third part of the extension is the music visualization component. The function of this extension is to create a visual representation of the songs, that the user adds to the player. It is based on the starter template, so there no changes in the structure, I just added more visualization options. I treated this template as a part of the bigger application, so it just completes the main extension. Code for each visualization consists of the name, the draw function that creates music visuals, and several helper functions, that it uses in the process.

-= 2 =-

I started only with only a vague idea of what I want to get in the end, and for some time after the start, I mostly just tested things without a real plan. But after some time, I decided to create a music player, with visualization functionality and scheduled a simple plan. I separated my work into three parts: the first one is about the player itself and all its components, the second part is the visualization module, and the last part is testing, bug-fixing, and polishing. When I worked on the midterm, I created the Gannt diagram and planned all the works that I expected. After the midterm, I have followed this schedule very closely and have stuck to it until the end. Luckily, I had enough time for almost every task and even had a few extra days to spend on the final visualization extension. The first two weeks of January I spent on the timeline and small display screen. This part gave me some trouble at the start. But in the end, I was able to complete everything on time. The last week of January I polished HTML and CSS, to make the player look a little more presentable. After that, I spent a week completing all suggested extensions and tweaked some of them to add a little flavor. Last two weeks, I worked on my visualizations. I had some difficulties with the implementation of the "rhythm game" idea, so when the time had come, I decided to ditch this idea and stick to the plan.

-= 3 =-

It is easy to say what I should be doing differently when you already know how to do things, but in the end, I think I should have spent more time working on the plan before starting to code something. I should have started with basic ideas of what I want my application to do, like what buttons I want and what they should do. Then I should have thought about all interactions beforehand. I had to improvise a lot when I understood that interaction between different functions caused wrong behavior. If I planned all these cross interactions from the start, I think I may structure my extensions differently. In the end, my plan was flawed, but at least I was able to stick to it and perceive through.

As I already stated, this application has lots of tricky interactions, and I fixed quite some bugs in the process. But to be safe, I sent my final project to my friends, asking them to evaluate this project. They helped me to find few errors that I missed myself. Most of these errors were caused by the timeline, because it did not trigger the normal playCycle function, but used the p5 jump method instead. I added the settings load function to the timeline and thus fixed most of these bugs. Also, my testers made few suggestions about the interface I also used to improve the user experience. In general, testers gave a good review of the project, and I'm very grateful to them.

[GameDev.net -- Beat Detection Algorithms](http://archive.gamedev.net/archive/reference/programming/features/beatdetection/index.html)    
[Visualizing Music with p5.js](https://therewasaguy.github.io/p5-music-viz/)    
[Beat Detection Using JavaScript and the Web Audio API | Beatport Engineering](http://joesul.li/van/beat-detection-using-web-audio/)   
[Getting Started with Web Audio API - HTML5 Rocks](https://www.html5rocks.com/en/tutorials/webaudio/intro/)   
[Read files in JavaScript](https://web.dev/read-files/)

[Custom input field](https://osvaldas.info/smart-custom-file-input)

[CSS customization for the buttons and inputs](http://tympanus.net/codrops/2015/09/15/styling-customizing-file-inputs-smart-way)

[How pseudo 3D works](https://www.lexaloffle.com/bbs/?tid=35767)