Originally, I had started looking into the project as explained throughout a series of articles written by Fernando Doglio, who had written a multiplayer text-based game. It wasn’t until I had spent time digging into it and working with it that I realized that this was for a terminal-based project: as in, someone would run a command in a terminal to start it, and then someone else could join based on that. I became worried that I wouldn’t be able to make what I wanted to—but I forged onward a bit longer to see if I could adapt some part of it.

Taking further looks into the repo <https://github.com/deleteman/tardis-project-engine>, I managed to find bits that I could still use and manipulate into what I wanted. Mainly the concepts of a Command Parser (in the /lib/commandParser.js file) and the “Game Cartridge” (in the /cartridges/sample.json, for example). If I had more time to clean up my code further and test it to ensure the functionality remains, I would even do more with utilizing the cartridge more directly, whereas right now it’s a bit half-and-half between hardcoding it and not.

The Command Parser’s purpose is for what it sounds like: the user enters a command like “move north”, and it helps to interpret what to make of that versus what it would do with “eat food.” Mine is also in a separate file from the main JS file, as well as an Attack Parser. This also shapes from the Command Parser’s idea, only with just breaking down the attacks further.

The “Game Cartridge” is where all the information for the game is. The descriptions, the conditionals, the NPCs, and is brilliant for building on cleanly. For example, it was easy to add in my own conditionals at the beginning of development *and* the end of development (when I was taking into consideration the possibility of players “fainting.”)

The only other main outside resource I directly looked at was <https://css-tricks.com/value-bubbles-for-range-inputs/>, which was for one of the opening screens (as seen in the joiningGame function in my code). I wanted to display an output based on a selected range input, and I really liked the concept of this. Even if it isn’t exactly the same format as the shown sliders, I like the connection and continuity.

I would also just like to say that I believe my Computer Science background helped me immensely with not only this project, but the concepts we went over in class. AJAX and MongoDB ended up being my best friends for pulling this project off to the extent that I could, and I had been introduced to both, with utilizing the latter a lot in my CSE 312 class through Docker.

If I were to start it over again from scratch, or to keep working on it outside of this semester, some things I would change are: making a GitHub repo for it earlier (it would’ve been a generally better way of updating it after working on it locally and being able to reference back), splitting up the code more to make decluttering it easier (there are probably even more things than I know that could be trimmed down, and the main final.js file is much larger than a single file should be, at least in terms of what I’ve been exposed to in practice), utilizing the game cartridge more consistently, and making a message to notify when a player has fainted rather than just disabling there ability to do damage and rendering the message when all players have fainted that they’re back at their last safe spot.

I spent a lot of time both in figuring out the code and in writing the story as I wanted it to be portrayed, but the end result on the user’s side is something I am content with, especially after seeing others playtest it as well!