Carter Wilson
Pokedex Journal

- At the beginning of the project, Trung already had a good idea of what type of data he wanted to work with and where to get it from; that is, what type of Pokemon data and the PokeDex API. He did not have an idea for what he wanted the design or styling to look like, so I spent most of the beginning of the project working on a cohesive theme for the whole page.

- I decided to approach the theme with the idea of showing the data we wanted on playing cards.

- After Trung set up the javascript for the landing page to cache all the pokemon, I started to arrange all the data. I put everything onto cards and then decided to turn them into flipcards. I started by trying to pile a lot of CSS functionality on top of already existing work. This made it difficult to get the cards to work correctly and even harder to debug. This taught me a valuable lesson that I used later on in the project: for more complicated layouts and styling, the best way to approach it is to build a basic prototype in codepen. This way you can test out a lot of methods until you have the building blocks for what you're trying to do. Then, I would move the code to our codebase and start adding on to it.

- The flipcard functionality and scrolling on the back of the card was fairly difficult to get right. The animation is built into CSS, so that was not too difficult, but I had to put in some extra time to learn about the perspective attribute to make it work. It also required lots of attention to detail in terms of which CSS was applied to different layers of the card, as lots of nested HTML components were needed to make it work.

- For overall design, I decided to use a gray background with white cards, and make our accent color the pokemon blue color. I used that color as often as possible. For fonts, I wanted something resembling a comic book type font because of the subject matter, and Google fortunately had a couple of fonts that fit the bill really well. These were used by linking the HTML to Google's fonts collection.

- Trung initially wanted to add all the data for a move to the card actually holding the moves. However, we discussed it and made a decision to move all the move data to a separate page to avoid replicating too much data. After that, he generated a page for the moves and a page for regions, and I worked to organize and style the data they contained.

- At this point I made an attempt to prototype one of the pages as a React component in the hopes that we could pass down data as props to have more pages. However, it became apparent that to do so, the entire app would have to be rewritten React, which

we felt we were too deep into the project to do.

- Trung wanted to add some kind of functionality to be able to determine the multiplier for damage done by an attack by a specific type of pokemon on the given pokemon. I had an idea for how this could be done, so I took over its implementation. The API we used did not return any useful data for it aside from the Pokemon type, so it had to be done from scratch. There is information out there for how different types effect each other, so I decided to build a matrix based off of it. I could have done some Python magic to build the matrix, but it was small enough to hard code in just a few minutes. I wrote the code to use the matrix and generate the multipliers in Javascript. I eventually got it to work, but spent a long time debugging what I thought was a logic issue and turned out to be a typo in the matrix where I turned a '1' into a '14'.

- We used bar graphs in a couple of different places in the site. There were really no other charts you could make with the data, so I decided not to pursue a javascript chart library and instead went with CSS Flexbox to build the bar graphs. The size of the bars was set by passing in variables to the CSS code, which is supported by basic CSS.

- While I considered using Bootstrap initially, I soon found that CSS flexbox is sufficient for pretty much anything you want to do layout-wise. There are some more complex interactivity functions that Bootstrap would make easier (i.e., flip cards) but I found that the Bootstrap could be somewhat limiting if you do not want to do it exactly the way it is implemented. Overall, I was impressed with how much you can accomplish solely using CSS and HTML 5 with animations and transitions.

- The page showing the comparison between two Pokemon was initially a reach goal I had wanted, and I ended up having time to implement it. After all the styling was already in place from previous features, it was relatively easy to set up. There was one interesting challenge, though, in setting up input validation for the page. I wanted to use a form so that I could use the HML5 input checks (because it's really easy!). To do that, I had to put the inputs inside a form. To run the checks, there has to be a submit input type button in the form. I wanted to display the information upon clicking the submit button, but of course, the submit button actually submits, which reloads the page so the data only shows for a second. So, I learned that I needed to stop the default submit behavior, which is a method that is actually built in to Javascript.

- Aside from growing graph bars, which I wanted from the beginning, I decided to add animations and transitions to a lot of the elements after they were already built. I found that adding fade in effects to newly loaded data was easy to implement and added a nice look. Given more time I would like to implement animations to make cards expand downward as new data is added. However, I had a lot of trouble making this work so decided to back off on it given time constraints.