Development

HTML:

Building the HTML was by far the easiest part of the whole project. I started it by adding links to the script, CSS, JQuery, and the API files. Next, for each function I would only target a small group of elements. In this way I was able to easily group all of the related elements into common divs which I could then target later. There were a few main divs that I made that I would later use such as the #inputs (had all the buttons for the search and see save and such while also including the input box), the #info (the div in which all of the information on each card was shown on click, also had the save, add to deck, fight, and exit buttons), and the #selections and #battlefield (these took care of the battle function). Nonetheless, one area which took up a lot of unnecessary code was the <img> tags for the images of the cards. I was forced to manually add all of the tags in the HTML as the JQuery would not work properly later if it the tags had not been added on load. Nextly, I was able to create a mask div which helped me later be able to create popups in which the background was unable to be modified as the mask covered it. On the other hand I also added an <audio> tag for the part of the battle scene where I played the Pokemon battle theme. Having it been manually added to the HTML allowed for it to be very easy to switch the audio source. I was also able to make the header in this area while adding both a title and images to it. It was difficult to get all the positioning of the elements to line up nicely but I'll go into more depth on this in the CSS section. One element which took a little bit of research was the dropdowns. These required the cooperation of both the CSS and the HTML to make it able to cleanly display options.

```
<div  id="viewdecks" class="dropdown">
    <button id="seedeckdrop" class="dropbtn">See Deck</button>
    <div class="dropdown-content" id="first">
    </div>
```

Dropdown div

```
<div id="images">
    <img id="img1" class="animate battle">
    <img id="img2" class="animate battle">
    <img id="img3" class="animate battle">
    <img id="img4" class="animate battle">
    <img id="img5" class="animate battle">
```

Problematic HTML repetition allowing the JQuery selector to function

SCRIPT:

Most of the script was in the JQuery document onload function. Nonetheless, most of the functions which included the saving, displaying of cards were not in this section. There were multiple functions surrounding the display and saving of cards. Some of these were the series of addtodeck(), makedeck(), and displaydeck() functions. These are pretty self-explanatory as the

first allows one to add a card to a deck, the second to make a deck, and the last to display the deck. Otherwise there was myFunction() which took the results of the API search and processed them until I could both display the information pulled and store it using the .data() JQuery function. Additionally, there were many JQuery selector event handler which did a variety of actions. One function allowed me to display the information of a card onclick by pulling the stored data on the card and showing the most important information such as the name, moves, types, image, etc. On the other token, I made three different event handlers for the battle part of the project. The first allowed informed the user to select the two cards they wanted to play with when the “Have a Battle” button was clicked. The second set up the battlefield once the second card was selected. This function was also in charge of starting the music. Finally the last event handler changed the battle scene as the user chose the move they wanted to employ. This event handler modified the Pokemon’s healths and would play different songs depending on whether or not the user “won” the battle. Lastly, the most important event handler was the one in which the actual call was made to the server and the information was pulled. This function only ran when the enter key was pressed which allowed for a simpler search. The final function was that which was in charge of displaying the mask div when a popup was shown.

```
$("#name").keyup(function(e){
    $("#decknameandcreator").hide();
    var code= e.which;
    if(code==13){
        var name= $("#name").val();
        $(".battle").removeClass("battle").addClass("animate");
        $.ajax({
            url: "https://api.pokemontcg.io/v1/cards?name="+name,
            type: 'GET',
            crossDomain: true,
            dataType: 'json',
            success: function(result) {
                console.log(result);
                myFunction(result) },
            error: function() {
                alert('Failed!');
            }
        });
    }
});
```

The API request function

CSS:

The formatting focused a lot on positioning the elements on the page as some would need to be fixed (the pop-ups) while others needed to be positioned relatively (mostly the images), and finally many had to positioned absolutely (most of the battle elements). On the other hand I was able to make buttons have certain characteristics based off their classes which simplified the process and allowed to smoothly have adjacent buttons have similar styles. A problem that arose during the positioning also was the having to take into consideration the z-index of each element in order to have the most pertinent element be displayed in front.

Otherwise the rest of the formatting was just changing the colors of divs and making sure the text was the correct size.

```css
#info{
    border: double black 2px;
    position: fixed;
    left: 0;
    right: 0;
    margin-left: 270px;
    margin-right: 270px;
    z-index: 2;
    text-align: left;
    background-color: white;
    overflow: auto;
    min-height: 400px;
}
```

Example CSS formatting

JQUERY ANIMATIONS:

One of the largest issues throughout the project was being able to able to get JQuery functions to work as many times the selector would have trouble identifying dynamically added elements. One way I was able to circumnavigate this problem was through the manual insertion of elements into the HTML file, which although it was a lot more writing, allowed the JQuery selector to work efficiently. Additionally a large part of the JQuery was the use of the .hide() and .show() functions which allowed me to display only the desired information and images. Otherwise I animated the images by making them shift up on mouseover. This was not very complicated yet required a duo of functions, one to shift up on mouseover and the other to shift down on mouseleave. Throughout the project, the JQuery selector targeted element tags, ids, and classes. This enabled for a multitude of animations to be running simultaneously.

JQuery image animations