**Hunt Project 1 Documentation**

* **My sites purpose**

This app is meant to be a card/stat-based multiplayer online battle app. A user can login, make a character that they think will perform well, then battle other people’s created characters either in a regular fight, or a death match. A death match will destroy the loser from the server, and increase a random stat from the winner by 1.

* **What part of the app does the API handle?**

The API handles both the battle logs and the list of fighters (each fighter is indexed by the fighter name, because all fighter names are unique). The client can request the fighters in order to display them, or when a user requests to fight with them. The client can then send back information to the server, and the server will update the fighters.

* **What went right and what went wrong?**

The overall execution of the project was great! The interaction especially went well, and I spent a long time on (the ability to select two fighters from the server, and fight with those given fighters). The algorithm for the battles is also pretty well done, but imbalanced in a lot of ways.

Some things that went wrong. The stats are somewhat imbalanced. Armor is just strictly better than health. Not sure how crit/speed stand. I’d have to experiment a little more. Crit seems very random though. If you crit, its usually an insta-win, which is very rng-y. The UI is sort of a mixed bag. It’s somewhat effective but also feels wrong too. The best part is definitely the display of the cards, and the battle log (they both scroll by themselves!! I didn’t know I could do that at first!!). But the field for creating a new fighter seems off. A lot better because of the flex layout.

* **If you were to continue, what would you do to improve your app?**

Right now, I have detailed info in the console to each battle you have, but I would love to add detailed information about the battle in the actual battle log itself. Like if you hover/click on the fight log, it’ll pull up a little bot that gives info about the fight that happened.

Better balance for the fights would definitely be beneficial.

A way to decide what stats your character has a greater chance of increasing if it wins a death match.

Better UI.

Unfortunately I wasn’t able to figure out how to stringify the server’s list of characters. I would definitely love to figure that out, but I ran out of time for now.

* **How did you go above and beyond?**

Referring to server data from the client is a very difficult process I learned. After lots of brainstorming I decided to just refer to the fighter objects as strings in an array called “active.” (Active can only hold 2 fighters since more than 2 fighters can’t fight. That’d be a cool though). When the client draws out each JSON character into its own div/card box, if the character’s name is the same as one of the character names in the active array, it adds another id to the div called ‘active’ so it draws in a different color. Then when ‘fighting,’ the client asks the server if two fighters with those specific fighter names exist. If so, the server sends back those JSON objects, the client runs the battle, then sends back the updated characters.

The battle log was a semi last minute decision actually. I figured there should be some way of seeing the history of the fighters/battles so I added it. Essentially it’s the same as a chat box, but the client sends automatic messages instead of users. In order to make it a chat box all I’d have to do is make a text box, and feed the user’s input into the ‘add log’ button. But that’s not the point of this project.

I also added a form of verification to the project. When creating a character, none of the stats can be higher than 15, but when a death match is win, a stat can upgrade from there. Problem was, I was using the same function server side to update it. Instead of adding another function server side, I decided to add some form of verification from the client side. Obviously this can be exploited, but when updating a character from a fight, the form sent back to the server includes an extra value ‘secure=${true}’. The server will check to see if there is a property on the object called true, and if there is, it’ll bypass the restrictions for the point values being maxed at 15 (and total point values being maxed at 36).

I also learned a lot of new CSS features such as display:inline-block to align the cards, and display:flex for the stat forms. Also using transform: scale(x); can increase the size of an element without affecting other elements spacing around it even if their spacing is relative to the element being transformed.  
I tried to spend a good amount of time making the UI good feedback, and make it satisfying to use, even though its not great.