Pokemon Battle Simulator:

Team Members - Drew Sheppard, Tharid Uralwong, Alexander Pena

Overview:

Our final project is a Pokemon battle simulator based on the web game Pokemon Showdown. It is a multiplayer game in which the player can battle against a friend using a randomly generated team with randomly generated moves, based loosely off of the 'Random Battle' format in Pokemon Showdown. Match-making is conducted by two players typing in each others' username. The game is also based off of Generation I, and so contains only the original 151 Pokemon as well as no items or abilities so as to faithfully reflect it.

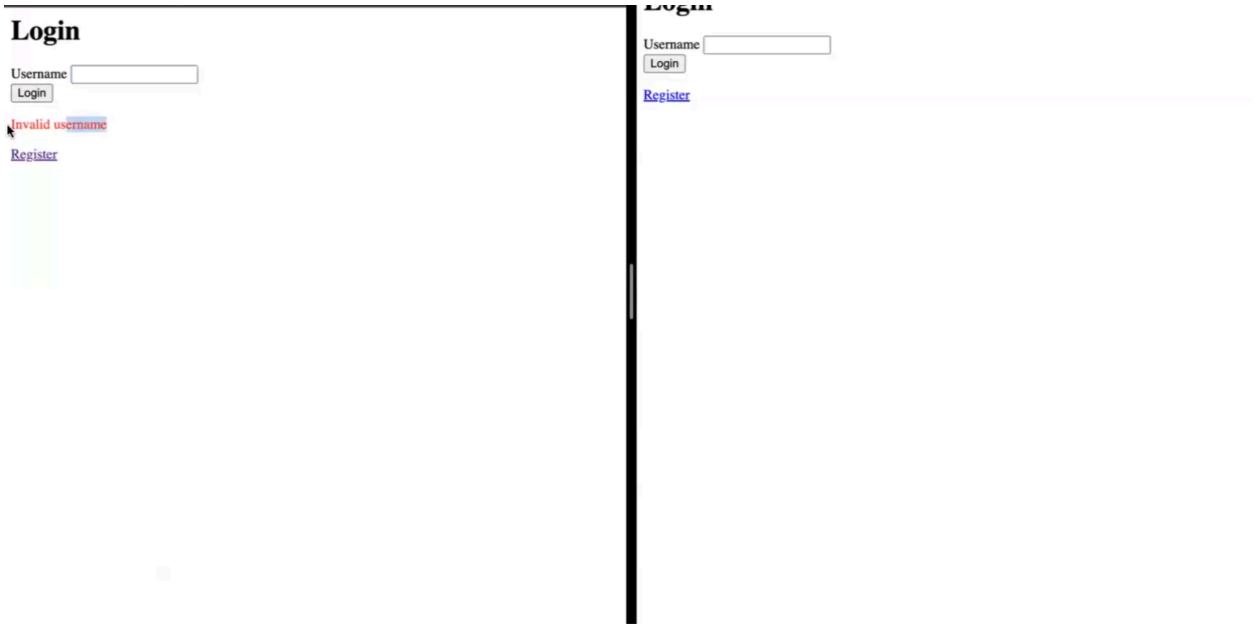
Pokemon in the battle are displayed as 2D sprites, with the player's pokemon always on the left side of the screen. As in a normal game of Pokemon, the winner is the first player to knock out all Pokemon on the opponent's team, after which the game will end and the player will be sent back to the main menu where they may look for another game.

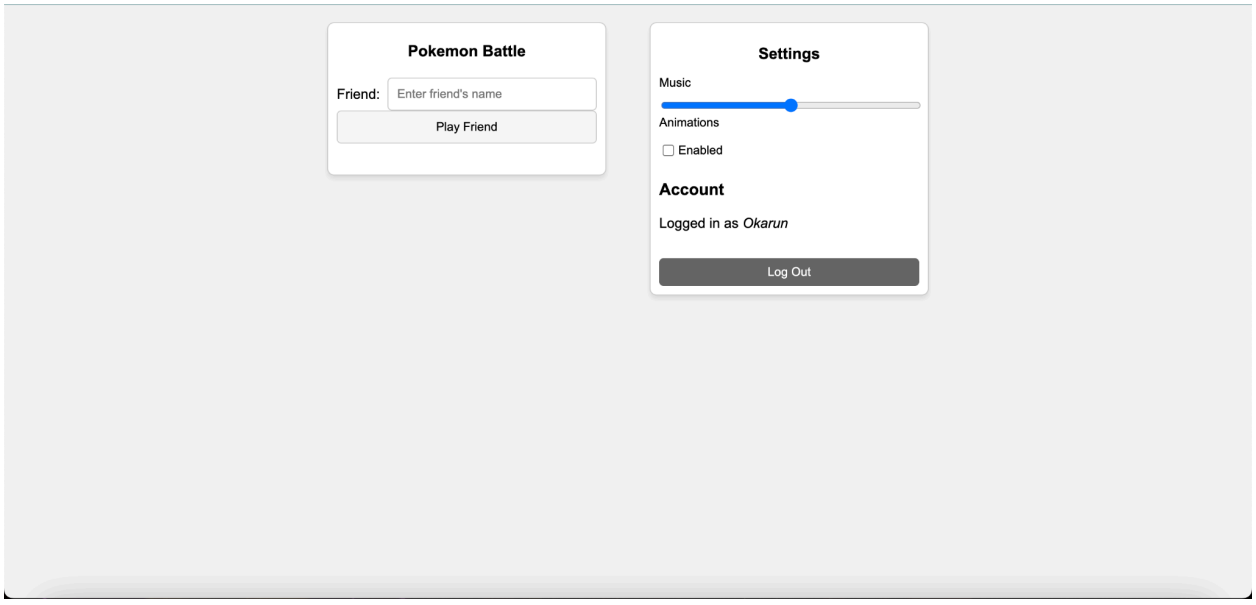
Front End:

The front end of our project starts with a login menu where the player may login with their username, or if they have not already created a user, may register an account. Login does not require a password for simplicity. After logging in, the player is redirected to the main menu where they may search for a battle. When the player has found a battle, they are taken to the battle page which contains 4 move buttons as well as the Pokemon they can choose to switch between. There is also a fully functional chat log that updates in real time.

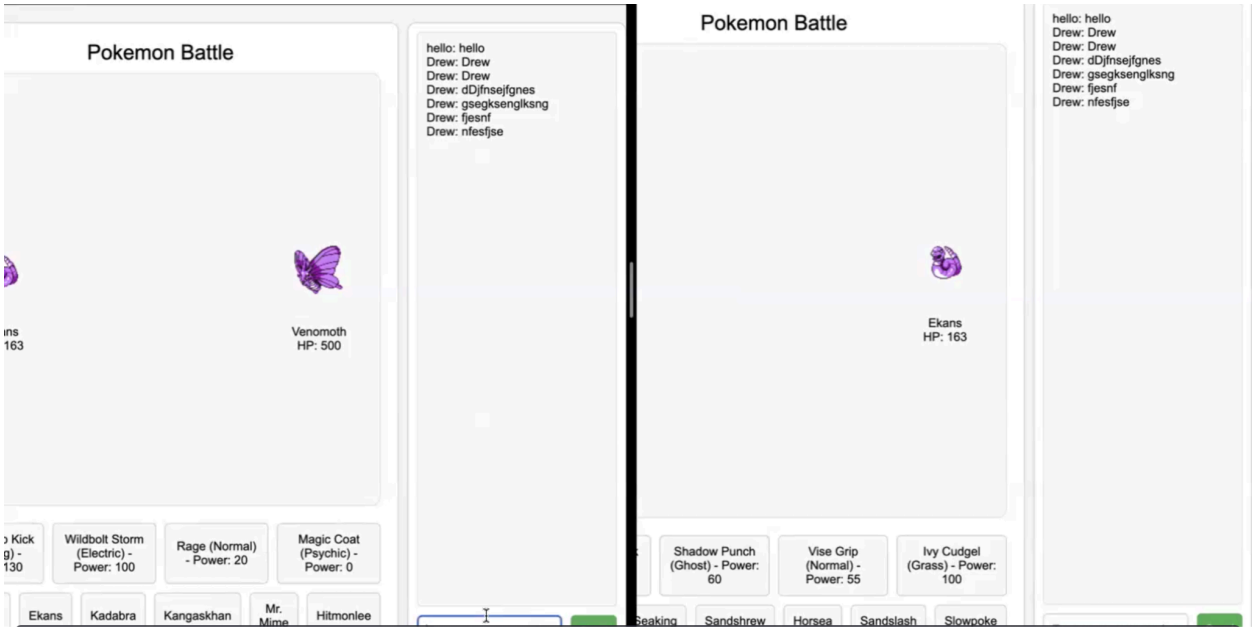
Screenshots

Login & Main Menu





Battle


Pokemon Battle
hello: hello
Drew: Drew
Drew: Drew
Drew: dDjfnsejfgnes
Drew: gsegksenglksng
Drew: fjesnf
Drew: nfesfjse
Venomoth
HP: 500
Wildbolt Storm (Electric) - Power: 100
Rage (Normal) - Power: 20
Magic Coat (Psychic) - Power: 0
Ekans
Kadabra
Kangaskhan
Mr. Mime
Hitmonlee
Pokemon Battle
Ekans
HP: 163
Shadow Punch (Ghost) - Power: 60
Vise Grip (Normal) - Power: 55
Ivy Cudgel (Grass) - Power: 100
Sandshrew
Horsea
Sandslash
Slowpoke
hello: hello
Drew: Drew
Drew: Drew
Drew: dDjfnsejfgnes
Drew: gsegksenglksng
Drew: fjesnf
Drew: nfesfjse

Backend

Our backend was implemented using a combination of Node.js, Express, and Socket.io. Though we originally planned to have the front-end mainly as a way to manage the backend where “the real application” would reside, our project ended up incorporating a fair amount of important manipulation code on both sides. We used MongoDB as our database to store information about pokemon, moves, and users. As mentioned, we use npm modules mongoose for storing data and express for creating routes. Socket.io and socket.io client are used to create a websocket that allows for real time updates, both for communicating information about moves and pokemon, as well as updating the chat log. For user information, the website uses routes to register users and store their information in the Mongo database. Battles have routes for getting the pokemon, getting the moves, and navigating to the correct battle page.

Routes:

GET/

GET /login

POST /login

GET /register

POST /register

POST /findBattle

POST /battleScene

POST /getRandomMoves

POST /getRandomPokemon

Database structure:

- **Pokemon schema**: { id: int, num: int, name: String, types: String[], baseStats, int[], heightm: float, weightkg: float, color: String, evos: “String”, }
- **Move schema**: { name: String, type: String, basePower: int }
- **Trainer schema**: {username: String }

Reflections

I would say that there was more that went wrong with the project than went right. For the things that went positive, I think that everyone was able to get together and sprint really well towards the end, and that showed a lot of commitment to the project and towards each other as a group. We were also able to get the two players connected using a mix of Express and Socket.io in a surprisingly small amount of time. Coming into the project, I assumed that this would be one of the biggest challenges we would have to face, and having it taken care of relatively early was a massive relief. Another strong point for our project is the chat system also works very well. Two players in a room are able to communicate with each other in real time, and this is a very good feature in my opinion - if nothing else, our website can be used as a real-time messaging service.

There are also many spots where I would change what we did. For me personally, I wanted to take the time to make different classes for the moves, pokemon, players, battles, and so on, and then build our app around those classes in a rational system. I failed both to communicate what I wanted and also decide the specifics of how to implement it in a reasonable amount of time, however, and so ended up writing a lot of code that while being well designed in my opinion, was completely out of sync with the real progress that my teammates had made and never made it into the final project. I think that I should have encouraged us to meet in person, reserve a study room, and draw a big XML diagram detailing how our website was going to work, and then have everyone work on a certain part.

Another spot where I think we had significant room for improvement was time management. I strongly suspect that we are not the only team to deal with this, but we ended up falling behind schedule and putting off work for one reason or another. In the end, we had to rush

to get the project done. This issue is so common in school projects in general that I do not believe there is any need for further elaboration.

The final spot where I think that things could have gone better is with understanding of the code. Though I took the time before writing anything to build a project with socket.io so as to gain an understanding of what to do, I did not express or mongoose, and the result was that I could only understand a portion of what my teammates wrote. We all ended up having to learn the different modules as we went, and I think that if we had gone in with a greater understanding of the system.