Pokedex task

Table of contents:

Tech Requirements

- Implement a Pokemon interface with Enums.
- Utilize React for UI development with use-query for state management.
- Implement page navigation using React Router.

General Components

- Implement a navigation bar for seamless page switching (reference Figma design).
- Design the following pages:
  - All Pokemons
  - My Pokemons
  - Fight

All Pokemons - Main Page

- Should be able to sort by
  - Name - asc & desc
  - Power - asc & desc
  - HP - asc & desc
- Should fetch all pokemons in a table with pagination (Figma)
- Should be able to search in the table - search should work with debounce
- If no pokemons are found in search it should display an empty state (Figma)
- Clicking on a table item should open a popup with its pokemon information (Figma)
- Owned Pokemons should have a small pokeball next to their name
- There should be a toggle button to switch between “My pokemons” and “All pokemons”
- Add cards view.

Fight Simulator

- Should be able to choose which of the pokemon that you have catched will fight in the battle (Figma)
- You cannot fight against pokemons you own.
- The pokemons with more speed attacks first
- The fight happen in turns - each turn one pokemon attacks while the other defends
- A defending pokemon takes damage based on the difference between the attacker “attack” points and the defender “defense” points
- A pokemon can be catched during a fight (Catch Mechanism)
- The fight is ended when
  - The “HP” (health points) of one of the pokemons reaches to zero
  - The opponent pokemon has been catched successfully (Catch Mechanism)
  - The opponent pokemon has not been catched and the maximum number of tries to catch a pokemon has been reached (Catch Mechanism)
- Add animation effects to enhance the visual experience during battles in the fight simulator.

Catch Mechanism

Determine a catch rate based on the Pokémon's current hit points and other factors.
Example: Set a catch rate of 10%. If the Pokémon's hit points fall below 20% of its maximum hit points, the catch rate increases to 20%.

Determine a maximum number of tries to catch a pokemon.
Example: if maximum number of tries is 3:

- Succeeding on the second try - pokemon is catched and added to the list of owned pokemons
- Succeeding on the third try - pokemon is catched and added to the list of owned pokemons
- Failing on the third try - the pokemon has "fled" and the fight is over

Random Mechanism

Use randomness to introduce variability and excitement into the battles.
For attack power, you could multiply the base attack power by a random factor within a certain range to simulate variability in the effectiveness of attacks.
Example: Multiply the base attack power by a random factor between 0.7 and 1.3.
For additional randomness, you could introduce random events during battles that affect Pokémon stats, such as temporary stat boosts or reductions, weather changes, or surprise appearances of other Pokémon.

Attack Miss Mechanism

Determine a miss chance based on the Pokémon's level, the attacker's accuracy, and other factors.
Example: Set a base miss chance of 5%. Adjust the miss chance based on factors such as the attacker's accuracy and the defender's evasion.

Type Advantages

Implement type advantages to add strategic depth to battles.
Example: Fire-type Pokémon should deal extra damage to Grass-type Pokémon. You could implement a 1.5x damage multiplier for attacks that have a type advantage.

Game Balance

Ensure that the game is balanced and fair to provide an enjoyable experience for players.
Test different values and mechanics to find the right balance between challenge and **accessibility.**
Adjust parameters based on player feedback and playtesting to refine the game design.
By incorporating these specific mechanics and values into your game design, you can create a Pokémon fight simulator that offers engaging battles with strategic depth and excitement. Adjustments can be made based on player feedback and testing to ensure a balanced and enjoyable gameplay experience.

Version 2.0

1. Authentication - with Cognito.
2. Fight mode with sockets - the same implementation applies, but in this situation, the fight will be between two users, instead of between one user and the computer (fight mode).
3. Unit tests with jest - shared component (UI).
4. Integration tests with jest - server side.
5. Docker + CI/CD

Figma: Pokédex figma
User story User story
Data JSON