Pokémon Battle Simulation MCP Server

This project is a technical assessment to create a Model Context Protocol (MCP) compliant server that provides Pokémon data and simulates battles. The server exposes two main functionalities: a comprehensive Pokémon data resource and a battle simulation tool, designed to be accessible to AI models and demonstrated through a dynamic web interface.

The project uses a Python **FastAPI** back-end to serve the data and a **HTML, CSS, and JavaScript** front-end to visualize the information and the animated battle sequences.

Features

- **Pokémon Data Resource**: An API endpoint that fetches and exposes detailed information about any Pokémon from the public PokeAPI.
  - Base stats (HP, Attack, Defense, etc.)
  - Types
  - Abilities
  - Available moves
- **Battle Simulation Tool**: An API endpoint that simulates a battle between any two Pokémon.
  - Implements core battle mechanics like type effectiveness, damage calculation, and turn order based on speed.
  - Returns a structured log of the battle, detailing every action.
- **Animated Web Interface**: A fun and creative front-end to demonstrate the server's capabilities, featuring:
  - Dynamic loading of Pokémon data.
  - An automated, animated battle sequence showing Pokémon fighting, taking damage, and fainting.
  - Sound effects for a more immersive experience.

Project Structure

The project consists of three main files:

- main.py: The Python back-end using the FastAPI framework. It handles all API requests, fetches data from PokeAPI, and runs the battle simulation logic.
- index.html: The main web page. It contains the structure of the application, including the Pokémon display cards, the battle stage, and control buttons.
- app.js: The front-end JavaScript. It handles all user interactions, communicates with the back-end API, and controls the animations and visual updates on the

webpage.

Setup and Installation

To run this project, you need Python 3.6+ installed. It is highly recommended to use a virtual environment to manage dependencies.

1. Create a Virtual Environment

Navigate to your project directory in the terminal and run the following commands:

```
# Create a virtual environment named 'venv'
python -m venv venv

# Activate the virtual environment
# On Windows:
venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate
```

2. Install Dependencies

With your virtual environment active, install the required Python packages using pip:

```
pip install fastapi "uvicorn[standard]" httpx
```

- **fastapi**: The web framework used for the server.
- **uvicorn**: The server that runs the FastAPI application.
- **httpx**: The HTTP client used to make requests to the PokeAPI.

How to Run the Application

The application requires two components to be running: the back-end server and the front-end interface.

1. Start the Back-End Server

With your virtual environment still active, run the following command in your terminal from the project's root directory:

```
uvicorn main:app --reload
```

The server will start and listen on http://127.0.0.1:8000. You should see a message like Uvicorn running on http://127.0.0.1:8000.

Important: You must leave this terminal window open. This is your running server.

2. Open the Front-End

Navigate to your project folder in your file explorer and double-click the index.html file. This will open the application in your default web browser.

The webpage will automatically connect to your running server to fetch Pokémon data.

How to Use the Web Interface

1. **Select Pokémon**: Use the two dropdown menus to choose the Pokémon you want to battle. The information cards will automatically update.
2. **Simulate Battle**: Click the **"Simulate Battle"** button. The application will contact the server to run the full battle simulation.
3. **Watch the Fight**: Once the simulation is complete, the **"Auto Play"** button will become active. Click it to watch the animated battle unfold on the Battle Stage.
4. **Reset**: Click the **"Reset"** button to clear the battle and start a new one.

API Endpoints (for LLM/Tool Use)

The server exposes two main endpoints as per the assessment requirements.

1. Pokémon Data Resource

This endpoint provides detailed data for a specific Pokémon.

- **Endpoint**: GET /pokemon/{name}
- **URL Parameter**:
    - name (string): The name of the Pokémon (e.g., "pikachu", "charizard").
- **Example Query (using curl)**:
  ```
  curl http://127.0.0.1:8000/pokemon/pikachu
  ```
- **Example Response**:
  ```
  {
    "name": "pikachu",
    "id": 25,
    "sprite":
  "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/25.p
  ```

```
ng",
  "types": ["electric"],
  "stats": {
    "hp": 35,
    "attack": 55,
    "defense": 40,
    "special-attack": 50,
    "special-defense": 50,
    "speed": 90
  },
  "abilities": [
    {"name": "static", "description": "Has a 30% chance of paralyzing attacking
Pokémon on contact."},
    {"name": "lightning-rod", "description": "Absorbs Electric-type moves, raising
Special Attack by one stage."}
  ],
  "moves": [
    {"name": "mega-punch", "type": "normal", "power": 80, ...},
    {"name": "pay-day", "type": "normal", "power": 40, ...}
  ]
}
```

2. Battle Simulation Tool

This endpoint simulates a battle between two Pokémon and returns a structured log of the events.

- **Endpoint**: GET /battle/simulate
- **Query Parameters**:
  - pokemon1 (string): The name of the first Pokémon.
  - pokemon2 (string): The name of the second Pokémon.
- **Example Query (using curl)**:
```
curl
"http://127.0.0.1:8000/battle/simulate?pokemon1=pikachu&pokemon2=bulbasaur"
```

- **Example Response**:
```
{
  "battle_log": [
    {
      "action": "attack",
      "attacker": {"name": "pikachu"},
      "defender": {"name": "bulbasaur", "hp_left": 30},
      "move": "quick-attack",
      "damage": 15,
      "text": "Pikachu used Quick-attack dealing 15 damage!"
    },
    {
      "action": "attack",
      "attacker": {"name": "bulbasaur"},
      "defender": {"name": "pikachu", "hp_left": 22},
      "move": "tackle",
      "damage": 13,
      "text": "Bulbasaur used Tackle dealing 13 damage!"
    },
    {
      "action": "faint",
      "pokemon": {"name": "pikachu"},
      "text": "Pikachu fainted!"
    },
    {
      "action": "end",
      "text": "Battle Over. Winner: Bulbasaur"
    }
  ]
}
```