# Beyond the Cards: An Application to Pokemon TCG Database

# ## Team Members

Chengyi Li

Lewei Lin

Zhanyang Sun

# ## Installation & Usage

\*\*\*Please ensure you have access to a stable Wi-Fi connection before using our app. If you do not have Wifi access, please uncheck “show images” in the checkbox shown at the beginning of the interface after executing main.py.\*\*\*

1. **Please make sure your *MongoDB* is properly installed. Login as localhost: 27017. Make sure your *MongoDB* database is empty before starting.**
2. **Download all the files, and move it to your desired workspace.**
3. **Navigate to the *demo* directory in your Terminal:**

cd demo

1. **Install dependencies:**

pip install -r requirements.txt

1. **Run *project.py*. You should see the below printout if data insertion is successful:**

…

Finished inserting for Pokemon\_Card

Finished inserting for Pokemon

Finished inserting for Attack

Finished inserting for Type

Finished inserting for Trainer

1. **Go to *MongoDB* to verify there exists *db1 to db5*.**
2. **In your Terminal, execute the *main.py*:**

streamlit run main.py

1. **Log in to interact with our web application as:**

**user (only *search*)**

* username: **user**
* password: **Dsci-551**

**database manager (all access of general users, plus *insert & update*)**

* username: **database\_manager**
* password: **Dsci-551**

**admin (all access of general users and database managers, plus *deletion*)**

* username: **admin**
* password: **Dsci-551**

# ## File Structure

├── **demo/**

│ └── comments.txt # Store data for the user comment section

│ └── **data/** # Containing our JSON files for this project

│ └── Attack.json

│ └── Pokemon.json

│ └── Pokemon\_Card.json

│ └── Trainer.json

│ └── Type.json

│ └── decoration pictures/ # Illustrations for our UI

│ └── **main.py** **# Main script for Front-End web application and UI design**

│ └── **project.py** **# Main script for Back-End hash functions and queries**

│ └── requirements.txt # Install required libraries and dependencies for the project

├── Final Report.pdf # Documentation of the overall implementation of our project

├── **raw\_data/**

│ └── pokemon\_tcg\_data/ # Raw data for our *Pokemon\_Card* and *Attack* tables

│ └── get\_json.py # Supplementary script for creating our raw JSON files

│ └── pokemon\_height\_weight.csv # Raw data for our *Pokemon* table

│ └── pokemon\_trainers.csv # Raw data for our *Trainer* table

│ └── pokemon\_type\_effetiveness\_chart.xlsx # Raw data for our *Type* table

│ └── web\_scrap\_height\_weight.py # Supplementary script for scraping & cleaning

└── **README.docx** **# Description of directories and what each file does.**

# ## Our Raw Data (Supplementary Notes)

*The raw data files are stored in the “****raw\_data****” folder.*

*The script for creating JSONs for our project is* ***get\_json.py*** in the “***raw\_data***” folder

*\*\*\*You don’t need to execute* ***get\_json.py*** *as it is only used for preliminary preparation, creating the JSON files for our project.\**\*\*

There are 5 JSON files used for this project, stored in the ***“data”*** folder under the “***demo***” directory:

* Attack.json
* Pokemon.json
* Pokemon\_Card.json
* Trainer.json
* Type.json

Below are descriptions of our collection schema and process for data cleaning and JSON creation:

**Type.json**

* **Schema:** 
  + Type\_ID # PRIMARY KEY int
  + Type\_name # str
  + Super\_effective\_against # list (of Type\_IDs)
  + Not\_effective\_against # list (of Type\_IDs)
  + No\_effect # list (of Type\_IDs)
  + Weakness # list (of Type\_IDs)
* The information was gathered from <https://pokemondb.net/type>, we made some adjustments and scraped it into an Excel file called *“pokemon\_type\_effectiveness\_chart.xlsx”*, including necessary information for *Type\_ID*, *Type\_Name*, *Super\_Effective\_Against*, *Not\_Very\_Effective\_Against*, *No\_Effect*, and *Weakness*. We wrote a script and converted this Excel into **Type.json**. In particular, we use the corresponding *Type\_ID*s to represent their *Type\_Name*s for easier search later on. A sample entry is as follows:

{"Type\_ID": 1, "Type\_name": "Normal", "Super\_effective\_against": [], "Not\_effective\_against": [13, 17], "No\_effect": [14], "Weakness": [7]},

**Trainer.json**

* **Schema:**
  + Trainer\_ID # PRIMARY KEY int
  + Trainer\_name # str
  + Pokemon\_team # str
  + Badges\_obtained # int
  + Region # list (of different regions)
* The information was gathered from [https://docs.google.com/spreadsheets/d/1GvuxjOBSZchMM4 q5z3lDGw0KLOSY3viEDI6HqDxsnDA/edit#gid=0](https://docs.google.com/spreadsheets/d/1GvuxjOBSZchMM4q5z3lDGw0KLOSY3viEDI6HqDxsnDA/edit#gid=0) and scraped into a CSV file called *“****pokemon\_trainers.csv****,”* including relevant information needed for *Trainer\_ID*, *Trainer\_Name*, and *Pokemon\_team*. Although some Trainers have the same name, we applied unique *Trainer\_ID*s to distinguish them. For *Badges\_obtained*, we randomly insert a number between 1-5 for each Trainer. As for *Region*, we learned from <https://pokemondb.net/location> that there are 10 major regions in the Pokemon world: [Kanto, Johto, Hoenn, Sinnoh, Unova, Kalos, Alola, Galar, Hisui, Paldea]. We would also randomly insert 1-5 regions for each Trainer for testing purposes. A sample entry is as follows:

{"Trainer\_ID": 2, "Trainer\_name": "Acerola", "Pokemon\_team": "USUM Elite Four rematch", "Badges\_obtained": 4, "Region": ["Hisui", "Kanto", "Kalos"]},

**Pokemon.json**

* **Schema:**
  + Pokemon\_ID # PRIMARY KEY int
  + Pokemon\_Name # str
  + Height # float
  + Weight # float
  + BMI # float
  + Type\_ID # FOREIGN KEY from *Type*  list
  + Trainer\_ID # FOREIGN KEY from *Trainer*  list
  + Region # list (of different regions)
  + img\_link # str
* The raw data information was scraped from <https://pokemondb.net/pokedex/stats/height-weight> and saved as *“****pokemon\_height\_weight.csv****”*. We initially excluded the duplicating Pokemons, and we had in total of 1025 unique Pokemons. When converting the CSV file to JSON, we similarly used *Type\_ID* from the Type Excel to represent their types. For testing purposes, 1-5 Regions were randomly inserted for each Pokemon. For Pokemon’s Trainer, we also randomly picked 1-6 numbers from 1-269 (*Trainer\_ID*s) for each Pokemon. A sample entry is as follows:

{"Pokemon\_ID": 1024, "Pokemon\_Name": "Terapagos", "Height": 0.2, "Weight": 14.3, "BMI": 6.5, "Type\_ID": [1], "Trainer\_ID": [137, 228, 229, 105], "Region": ["Hoenn", "Galar", "Kalos", "Hisui"], "img\_link": "<https://img.pokemondb.net/sprites/scarlet-violet/icon/terapagos-normal.png>"},

**Pokemon\_Card.json**

* **Schema:** 
  + Card\_ID # PRIMARY KEY int
  + ID # str (detailed ID description)
  + Pokemon\_ID # FOREIGN KEY from *Pokemon* list
  + HP # str
  + Type\_ID # FOREIGN KEY from *Type* list
  + Illustrator # str
  + Img\_path # str
  + Weakness # list (list of key-value pairs)
    - Type\_ID # list
    - Value # str
  + Resistance # list (list of key-value pairs)
    - Type\_ID # list
    - Value # str
  + Retreat\_cost # int
  + Attack\_ID # FOREIGN KEY from *Attack* list
* The raw data was pulled from a ***GitHub Pokemon TCG Database***: [***https://github.com/PokemonTCG/pokemon-tcg-data/tree/master/cards/en***](https://github.com/PokemonTCG/pokemon-tcg-data/tree/master/cards/en)
* First, we clone the data from GitHub to our local directory:

*# git clone* [*https://github.com/PokemonTCG/pokemon-tcg-data.git*](https://github.com/PokemonTCG/pokemon-tcg-data.git)

*# cd pokemon-tcg-data/cards/en*

* Then, we wrote a loop to loop over the 159 raw JSON files and retrieve the relevant information we need for our JSON. In particular, we applied each card with a unique *Card\_ID*, as well as gave each Attack a unique *Attack\_ID*. We again replaced the *Type\_Name* in the raw file with *Type\_ID* for our JSON. For Pokemon information, only the *Pokemon\_ID*s are foreign keys stored in this collection as references. A sample entry is as follows:

{"Card\_ID": 5, "ID": "base1-5", "Pokemon\_ID": [35], "HP": "40", "Type\_ID": [19], "Illustrator": "Ken Sugimori", "Img\_path": "<https://images.pokemontcg.io/base1/5.png>", "Weakness":[{"Type\_ID": 7, "Value": 2}], "Resistance":[{"Type\_ID": 11, "Value": -30}], "Retreat\_cost": 1, "Attack\_ID": [6, 7]}

**Attack.json**

* **Schema:**
  + Attack\_ID # PRIMARY KEY int
  + Attack\_name # str
  + Type\_ID # FOREIGN KEY from *Type*  list
  + Damage # str
  + Description # str
* The Attack information was separately retrieved when we created the ***Pokemon\_Card.json***. Therefore, only the corresponding *Attack\_ID*s were stored for each card’s Attack in ***Pokemon\_Card.json***. The Attack.json will serve as a reference if detailed information such as *Attack\_Name*, *Damage*, and *Description* are needed. A sample entry is as follows:

{"Attack\_ID": 24, "Attack\_name": "Thunder", "Type\_ID": [4, 4, 4, 19], "Damage": "60", "Description": "Flip a coin. If tails, Raichu does 30 damage to itself."}

**Other Details**

* “Lightning” type of Pokémon Cards was treated as an “Electric” type (Type\_ID: 4).
* We manually replaced all “Dark” with “Darkness”(Type\_ID: 16), “Steel” with “Metal” (Type\_ID: 17) in the raw *“pokemon\_height\_weight.csv”* and *“pokemon\_type\_effectiveness\_chart.xlsx”* data files after scraping the data from the websites, so that the *Type\_Name* could better match the “type” names in raw JSON for both Pokemon and Pokemon Cards.
* Not all attributes of each Entity have values. For example, a Pokemon card may not have Resistance.