**Technical Documentation for the Interactive SQL-Based Pokédex Capstone Project**

Hector Gerardo

Computer Information Systems Capstone Project

SUNY Polytechnic Institute

Introduction

The goal of this capstone project is to develop an interactive Pokédex application, powered by a PostgreSQL database, Python-based interface, and AI-assisted natural language querying using large language models (LLMs) through the Ollama API. A Pokédex, in the context of the Pokémon franchise, is a digital encyclopedia that catalogs and provides detailed information about various Pokémon species. This project transforms that concept into a fully functional, interactive digital application that allows users to access and explore this information through intuitive queries.

The purpose of this write-up is to offer a detailed technical guide that explains both the structure and logic behind the project and how the different components interact to provide a seamless user experience.

Data Acquisition

The primary dataset for this project was sourced from Kaggle.com, a popular online platform for datasets and machine learning competitions. The specific dataset, titled “Complete Pokémon Dataset,” contains comprehensive information on Pokémon from Generations 1 through 8. This includes attributes such as name, type, abilities, base stats, egg groups, generation, and combat effectiveness against other types.

A publicly available CSV file containing comprehensive data on all Pokémon from Generations 1 through 8. This dataset included attributes such as name, type, abilities, base stats, egg groups, generation, and combat effectiveness against other types.

Example raw data snippet:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **name** | **type1** | **type2** | **hp** | **attack** | **defense** | **abilities** | **classification** |
| Bulbasaur | grass | poison | 45 | 49 | 49 | ['Overgrow', 'Chlorophyll'] | Seed Pokémon |

Cleaning efforts included:

* Correcting a consistent misspelling of the "classification" column, originally labeled "classfication."
* Parsing stringified lists like abilities into actual Python lists using ast.literal\_eval.
* Replacing NaN or empty values with None before database insertion.
* Validating all keys referenced in many-to-many tables (e.g., checking that all ability/type values existed before relational inserts).

Example bug:

# This line incorrectly used 'classfication' (misspelled)  
cursor.execute("INSERT INTO pokedex (...) VALUES (..., row['classfication'], ...) ")

This was corrected by identifying the typo and adjusting both the database schema and insertion logic accordingly (see Databaserelational.txt).

Database Overview

The database schema was normalized and designed with scalability and relational integrity in mind. The major tables include:

* pokedex: stores core identifiers and classification details
* stats: stores numeric attributes like HP, attack, speed, etc.
* types and pokemon\_types: handle many-to-many type relationships
* abilities and pokemon\_abilities: handle many-to-many ability mappings
* physical\_info: stores metrics like height, weight, capture rate
* effectiveness: holds matchup values for each type against others

Below is a simplified version of the ER diagram used:

The motivation for separating stats, types, and abilities into distinct tables was to reduce redundancy and maintain consistency across relationships. Foreign key constraints and cascade delete actions ensure data integrity.

Additionally, lookup maps were created dynamically in Python (see type\_id\_map and ability\_id\_map) to avoid redundant insert queries.

Integration of LLMs (Large Language Models)

Large Language Models like Mistral or SQLCoder were integrated using the Ollama Python API (Project Charlie). LLMs were used to interpret natural language input from users and convert it into SQL queries that can be run against the database.

Key components:

* Prompting was structured with clear instruction-based formatting, e.g., "Convert this question into SQL given this schema."
* Schema and example queries were dynamically embedded in the prompt to guide the model.
* Debugging output (Project Delta) printed selected tables/columns and SQL queries for traceability.

Here’s a prompt example used:

prompt = f"""  
You are a helpful assistant that converts user questions into PostgreSQL SQL queries.  
Schema: {SCHEMA}  
Question: \"What is Pikachu's speed?\"  
"""

The app also reinterprets results in Pokédex style:

"Pikachu. Speed: 90. Type: Electric."  
 LLM outputs can be updated by simply changing the model identifier in the configuration:

MODEL\_NAME = 'sqlcoder' # Replaceable without logic refactor

Final Application Overview

Built in Python with Tkinter, the GUI includes:

* Entry box for free-text questions
* Pokédex narration of results
* Image rendering using PokéAPI
* Scrollable result area

Example interactions:

* User input: "List all fire-type Pokémon"
* Output: List with names, types, base stats
* User input: "What is Bulbasaur's base happiness?"
* Output: "Bulbasaur. Base happiness: 70."
* Visual feedback includes Pokémon sprites pulled from the PokeAPI:

url = f'https://pokeapi.co/api/v2/pokemon/{pokemon\_name.lower()}'

Image Examples:



Future Work

This project has several planned improvements:

* Add evolution chain logic and support for mega forms
* Build a Flask web-based interface for remote access
* Improve SQL error handling with LLM fallback recovery
* Explore voice input using speech-to-text

Applications of this architecture extend beyond Pokémon:

* Museums: searchable artifact catalogs
* Medical: diagnosis lookup from patient records
* Education: natural-language access to textbook databases

Conclusion

This capstone project merges structured data systems with modern AI-driven interfaces to create an engaging and useful tool. It showcases a deep understanding of database normalization, data cleaning, and Python scripting, while also pushing into emerging technology with LLMs. By building a system that is modular, explainable, and user-friendly, the project offers a practical blueprint for future applications that bridge traditional computing and AI.

With enhancements, it has the potential to evolve into a highly generalized natural-language interface for any structured dataset.