POKE BATTLES

AIDI 2001 - 01

Final Project Report

Project by:
Michael Molnar (100806823)
Tejas Vyas (100809103)
Hibba Imtiaz (100794061)

In this Report

Introduction

Pokémon is world's third biggest video game franchise of all time [1]. Created by Satoshi Tajiri, in 1996 as a role-playing video game for the Game Boy handheld game console, this game franchise has earned over US$ 92 billion throughout the world and has been part of popular culture in various forms such as:

- Video Games
- Manga
- Anime
- Trading Card Game
- Merchandise
- Board Games
- Movies
- TV Shows

Expert systems today can be used in all skilled industries and can be very readily implemented in game design and engineering, so in this project our intention is to use Pokémon Trading Card game as an inspiration to create a new Expert System which can be used as an intelligent opponent.

Abstract

In this project we create a trading card battling system, named **PokeBattlES** inspired by Pokémon and popular media “memes”. The project uses Pyknow as a Knowledge Engine and Flask as the hosting web service. The expert system uses Rule-Based Expert System concepts and includes rules relating to:

- Minion Types handling interactions between types such as Fire, Water, Electric, Grass, Rock, and Legendary.
- Buffs and de-buffs handling interactions resulting from Energy Cards.
- Battle Damage handling including attack-attack, attack-defense, and defense-attack cases.
- Win conditions.
- Gameplay update scenario thresholds such as HP threshold to allow player handicap using legendary minions.

For the gameplay, users have ability to select a set of health points and can interact with the expert system to play the game and get explanations on how the system behaves through each turn receiving feedback relating to game progression.

Task and Purpose

The core task for this project is to create an expert system that can be used to simulate intelligent game play of trading card game. In order to achieve this task, we use the Python frameworks Pyknow and Flask to create an app where user can interact with the expert system in a turn-by-turn manner by clicking on the desired card.

Our purpose behind this project is to:

- Use the opportunity to create an expert system for gaming industry, covering general idea behind all trading card games.
- Create an enjoyable experience for the user to interact with the AI and provide necessary explanations.
- Make the game fun and nostalgic! We made the cards a meme version to go Plus Ultra!

Knowledge Sources and Acquisition

For this project we took a Pokémon dataset available on Kaggle [2]. The csv file contained all the information needed for the game logic, such as card number, name, type, attack and defense stats for each Pokémon. All these stats are required for the game and are integrated into the game logic.

We made some additions to the dataset, for instance, we incorporated the energy cards, which were not provided in the Kaggle dataset. Furthermore, some deductions were made in order to reduce the complexity of the game logic. We only kept the Pokémon’s that belonged to the first generation, the rest of the generations were not inserted into game and are not part of the gameplay. Moreover, the minion types of Pokémon were also reduced to 5 developers’ choice types, as mentioned earlier.

For the card creation process, we used a web tool PCM II [3] that develops cards for different minion types. The images used for these cards were taken from various meme websites. We took care not to cause copyright infringement of any kind.

Knowledge Design and Engineering

To remain within the syllabus of the course, we have used Pyknow [4] to design and engineer the logic and rules of the game. We have also utilized pandas and numpy libraries for various other tasks.

The game is divided into the following three sections that are interlinked together to make the game core:

1. **Facts**: This part of the game core holds all facts as variables that are regularly used throughout the game logic and rules, such as the player and computer HP stat or the hand dealt. The facts are called and retracted according to their usage.
2. **Knowledge Engine**: This is the main logic of the game core that deals with the gameplay. It contains the Pokémon dataset as pandas dataframe which is referenced by the rules every now and then.
3. **Rules**: Rules are introduced inside the game logic as functions, with parameters, which can be called by other rules. They are intertwined with one another and most of the time are fired in succession. Rules work with facts which are passed between the rules.
   There around 50 rules in our game core, which includes card dealing, threshold hits, type comparisons and many more.

Architecture and Implementation

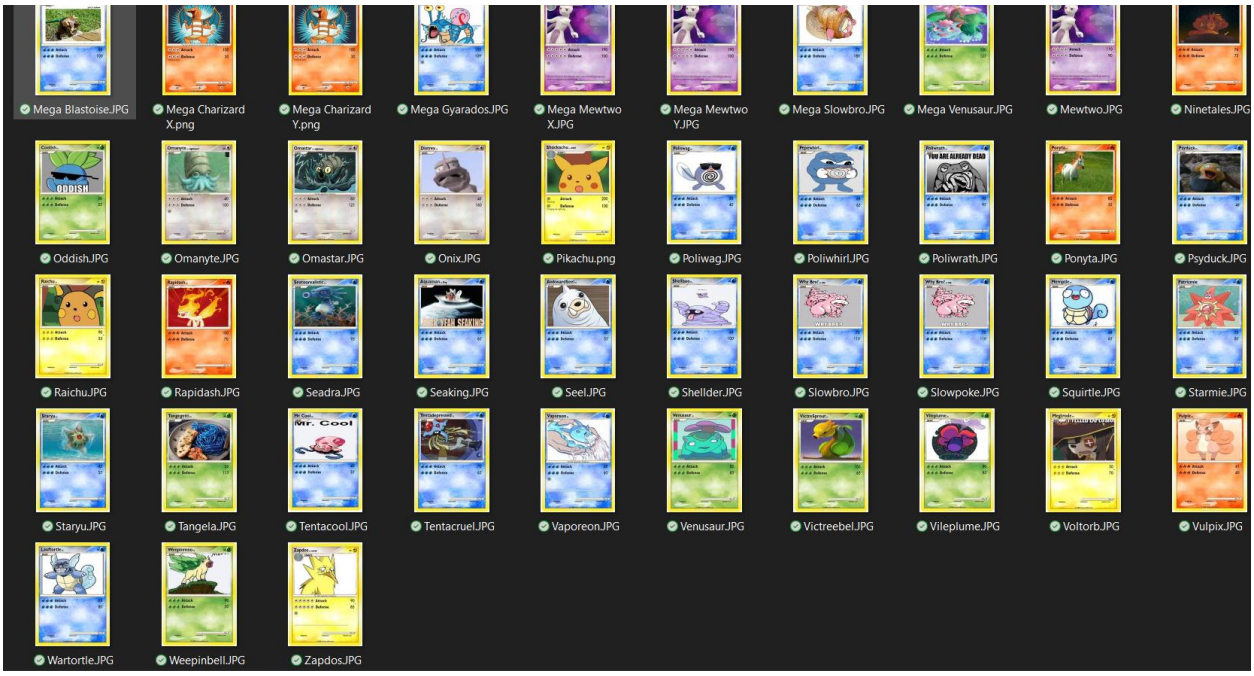
The project architecture includes 3 major components:

Data and Assets

As mentioned in Knowledge Sources, we retrieved data on the respective minions from a dataset publicly available on Kaggle. This dataset contains csv with the necessary details for the core expert system to be configured. It contains data to set up 92 cards including the base attack and defense, type, as well as additional features to make the expert system better in future iterations.

	A	B	C	D	E	F	G	H	I	J	K	L
1	#	Name	Type 1	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generatio	Legendary
2	1	Bulbasaur	Grass	318	45	49	49	65	65	45	1	FALSE
3	2	Ivysaur	Grass	405	60	62	63	80	80	60	1	FALSE
4	3	Venusaur	Grass	525	80	82	83	100	100	80	1	FALSE
5	151	Mega Ven	Grass	625	80	100	123	122	120	80	1	FALSE
6	4	Charmand	Fire	309	39	52	43	60	50	65	1	FALSE
7	5	Charmelec	Fire	405	58	64	58	80	65	80	1	FALSE
8	6	Charizard	Fire	534	78	84	78	109	85	100	1	FALSE
9	152	Mega Cha	Fire	634	78	130	111	130	85	100	1	FALSE
10	153	Mega Cha	Fire	634	78	104	78	159	115	100	1	FALSE
11	7	Squirtle	Water	314	44	48	65	50	64	43	1	FALSE
12	8	Wartortle	Water	405	59	63	80	65	80	58	1	FALSE
13	9	Blastoise	Water	530	79	83	100	85	105	78	1	FALSE
14	154	Mega Blas	Water	630	79	103	120	135	115	78	1	FALSE
15	25	Pikachu	Electric	320	35	55	40	50	50	90	1	FALSE
16	26	Raichu	Electric	485	60	90	55	90	80	110	1	FALSE
17	37	Vulpix	Fire	299	38	41	40	50	65	65	1	FALSE
18	38	Ninetales	Fire	505	73	76	75	81	100	100	1	FALSE
19	43	Oddish	Grass	320	45	50	55	75	65	30	1	FALSE
20	44	Gloom	Grass	395	60	65	70	85	75	40	1	FALSE
21	45	Vileplume	Grass	490	75	80	85	110	90	50	1	FALSE
22	54	Psyduck	Water	320	50	52	48	65	50	55	1	FALSE

In addition to the Data, 92 custom-made “meme” cards were created to preserve uniqueness and make the User Interface seem more friendly and enjoyable.



PyKnow Module

This module holds game logic and rules. The Rules govern the actual game play and act as a customized game engine which can handle all game-play scenarios using pyknow library and python logic.

The logic is divided into Fact Subclasses, a Knowledge Engine class as well as a driver method which allows Flask App to update the state of the Expert System RESTfully.

```
53    ''' Water beats Fire
54        Electric beats Water
55        Grass beats Rock
56        Rock beats Electric
57        Fire beats Grass'''
58    # The Fact Subclasses
59    class PokeES(Fact):...
64    class RegularPokeCards(Fact):...
73    class AllPokeCards(Fact):...
82    class ComputerAttackDifficulty(Fact):...
92    class DealtCards(Fact):...
99    class UserCards(Fact):...
104   class ComputerCards(Fact):...
109   class HP(Fact):...
116   class LegendaryThreshold(Fact):...
122   class LegendaryRounds(Fact):...
130   class RoundNumber(Fact):...
136   class UserCard(Fact):...
143   class UserCardType(Fact):...
149   class ComputerCard(Fact):...
154   class ComputerCardTypes(Fact):...
159   class ComputerCardType(Fact):...
164   class UserEnergyMultiplier(Fact):...
171   class ComputerEnergyMultiplier(Fact):...
178   class WhoGetsMultiplier(Fact):...
184   class Multiplier(Fact):...
190   # The Knowledge Engine
191   class PlayPokeES(KnowledgeEngine):...
644   # Helper method to Instantiate Pyknow
645   def runIt(hp_val, ai_val, uc, ac, rn, selectedid, state, thp, lrn):...
```

When the expert system is running, the following 41 rules are triggered allowing the knowledge engine to function.

```
[
    Rule(PokeES('battle'), UserCard(W('u_card')), ComputerCard(W('c_card')), WhoGetsMultiplier(W('bonus')),
        Multiplier(W('multiplier')), UserEnergyMultiplier(W('ueng')), ComputerEnergyMultiplier(W('ceng')),
        HP(user=W('uhp'), computer=W('chp')), RoundNumber(round_num=W('rnum'))) => <function PlayPokeES.battle at 0x000002A8E659D160>,
    Rule(PokeES('both_legendary'), UserCardType(W('u_type')), ComputerCardType(W('c_type'))) => <function PlayPokeES.both_legendary at 0x000002A8E658EEE0>,
    Rule(PokeES('check_deal'), DealtCards(W('dealt_cards'))) => <function PlayPokeES.check_deal at 0x000002A8E6547F70>,
    Rule(PokeES('check_deal_leg'), DealtCards(W('dealt_cards'))) => <function PlayPokeES.check_deal_leg at 0x000002A8E658E160>,
    Rule(PokeES('comp_attack_optimal'), ComputerCards(W('c_cards'))) => <function PlayPokeES.comp_attack_optimal at 0x000002A8E658EC10>,
    Rule(PokeES('comp_attack_random'), ComputerCards(W('c_cards'))) => <function PlayPokeES.comp_attack_random at 0x000002A8E658EB80>,
    Rule(PokeES('comp_legendary'), UserCardType(W('u_type')), ComputerCardType(W('c_type'))) => <function PlayPokeES.comp_legendary at 0x000002A8E659D040>,
    Rule(PokeES('comp_play'), ComputerCard(W('c_card'))) => <function PlayPokeES.comp_play_second at 0x000002A8E658E940>,
    Rule(PokeES('comp_plays_no_legendary'), ComputerCards(W('c_cards'))) => <function PlayPokeES.comp_plays_first_no_legendary at 0x000002A8E658ECA0>,
    Rule(PokeES('comp_vs_electric'), ComputerCardTypes(W('c_types')), ComputerCards(W('c_cards'))) => <function PlayPokeES.comp_vs_electric at 0x000002A8E658E790>,
    Rule(PokeES('comp_vs_fire'), ComputerCardTypes(W('c_types')), ComputerCards(W('c_cards'))) => <function PlayPokeES.comp_vs_fire at 0x000002A8E658E5E0>,
    Rule(PokeES('comp_vs_grass'), ComputerCardTypes(W('c_types')), ComputerCards(W('c_cards'))) => <function PlayPokeES.comp_vs_grass at 0x000002A8E658E700>,
    Rule(PokeES('comp_vs_legendary'), ComputerCardTypes(W('c_types')), ComputerCards(W('c_cards'))) => <function PlayPokeES.comp_vs_legendary at 0x000002A8E658E8B0>,
    Rule(PokeES('comp_vs_rock'), ComputerCardTypes(W('c_types')), ComputerCards(W('c_cards'))) => <function PlayPokeES.comp_vs_rock at 0x000002A8E658E820>,
    Rule(PokeES('comp_vs_water'), ComputerCardTypes(W('c_types')), ComputerCards(W('c_cards'))) => <function PlayPokeES.comp_vs_water at 0x000002A8E658E670>,
    Rule(PokeES('compare_types'), UserCardType(W('u_type')), ComputerCard(W('c_card'))) => <function PlayPokeES.compare_types at 0x000002A8E658EE50>,
    Rule(PokeES('computer_attack_style'), ComputerAttackDifficulty(diff=W('c_diff'))) => <function PlayPokeES.computer_attack_style at 0x000002A8E658EAF0>,
    Rule(PokeES('computer_defense'), UserCardType(W('u_type')), ComputerCards(W('c_cards'))) => <function PlayPokeES.computer_defense at 0x000002A8E658E550>,
    Rule(PokeES('computer_first_energy'), ComputerCards(W('c_cards'))) => <function PlayPokeES.computer_first_energy at 0x000002A8E658EA60>,
    Rule(PokeES('computer_plays_first'), ComputerCards(W('c_cards'))) => <function PlayPokeES.computer_plays_first at 0x000002A8E658E9D0>,
    Rule(PokeES('computer_plays_second'), ComputerCards(W('c_cards'))) => <function PlayPokeES.computer_plays_second at 0x000002A8E658E430>,
    Rule(PokeES('computer_second_energy'), ComputerCards(W('c_cards'))) => <function PlayPokeES.computer_second_energy at 0x000002A8E658E4C0>,
    Rule(PokeES('deal'),) => <function PlayPokeES.deal_cards at 0x000002A8E6547EE0>,
    Rule(PokeES('deal_full_deck'),) => <function PlayPokeES.deal_full_deck at 0x000002A8E658E0D0>,
    Rule(NOT(PokeES(),), AllPokeCards(p_key=W('p_key'), p_name=W('name'), p_type=W('p_type'), p_attack=W('attack'),
                                      p_defense=W('defense'))) => <function PlayPokeES.def_all_poke_cards at 0x000002A8E6547B80>,
    Rule(NOT(PokeES(),), RegularPokeCards(p_key=W('p_key'), p_name=W('name'), p_type=W('p_type'), p_attack=W('attack'),
                                      p_defense=W('defense'))) => <function PlayPokeES.def_poke_cards at 0x000002A8E6547C10>,
    Rule(PokeES('end_game'), HP(user=W('uhp'), computer=W('chp'))) => <function PlayPokeES.end_the_game at 0x000002A8E659D280>,
    Rule() => <function PlayPokeES.new_game at 0x000002A8E6547CA0>,
    Rule(PokeES('next_round'), RoundNumber(round_num=W('rnum'))) => <function PlayPokeES.next_round at 0x000002A8E659D1F0>,
    Rule(PokeES('no_legendary'), UserCardType(W('u_type')), ComputerCardType(W('c_type'))) => <function PlayPokeES.no_legendary at 0x000002A8E659D0D0>,
    Rule(PokeES('show_scores'), HP(user=W('uhp'), computer=W('chp')), RoundNumber(round_num=W('rnum')),
                        LegendaryThreshold(W('legthresh'))) => <function PlayPokeES.print_current_scores at 0x000002A8E6547DC0>,
    Rule(PokeES('show_user_cards'), UserCards(W('u_cards'))) => <function PlayPokeES.show_user_cards at 0x000002A8E658E1F0>,
    Rule(PokeES('split_cards'), DealtCards(W('dealt_cards'))) => <function PlayPokeES.split_cards at 0x000002A8E658E040>,
    Rule(PokeES('threshold_hit'), LegendaryRounds(leg_rounds=W('lrounds')), HP(user=W('uhp'), computer=W('chp'))) => <function PlayPokeES.thresh_hit at 0x000002A8E6547D30>,
    Rule(PokeES('user_behind_legendary'),) => <function PlayPokeES.user_behind_legendary at 0x000002A8E6547E50>,
    Rule(PokeES('user_first_energy'), UserCards(W('u_cards'))) => <function PlayPokeES.user_first_energy at 0x000002A8E658E3A0>,
    Rule(PokeES('user_legendary'), UserCardType(W('u_type')), ComputerCardType(W('c_type'))) => <function PlayPokeES.user_legendary at 0x000002A8E658EF70>,
    Rule(PokeES('user_plays_first'), UserCards(W('u_cards'))) => <function PlayPokeES.user_plays_first at 0x000002A8E658E310>,
    Rule(PokeES('user_plays_second'), UserCards(W('u_cards'))) => <function PlayPokeES.user_plays_second at 0x000002A8E658ED30>,
    Rule(PokeES('user_second_energy'), UserCards(W('u_cards'))) => <function PlayPokeES.user_second_energy at 0x000002A8E658EDC0>,
    Rule(PokeES('whose_turn'), RoundNumber(round_num=W('rnum'))) => <function PlayPokeES.whose_turn at 0x000002A8E658E280>
```

More rules can be added easily, and existing rules can be broken into further categories to make the expert system more explainable.

Flask Web Application

Since the pyknow class is console only, we created a UI written in Flask framework to allow users to interact with the expert system and enrich gameplay further.

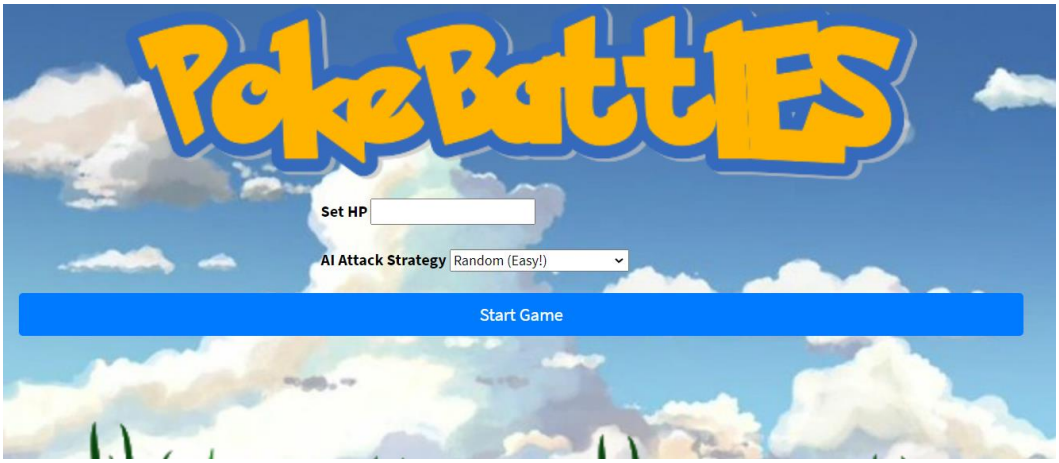
The app contains 5 main routes based on app state and is deployed on Azure [5]:

Home Page

- This is the landing page of the app, allowing users to get a quick idea about the game and redirecting them to game setup page when Begin Game button is clicked.

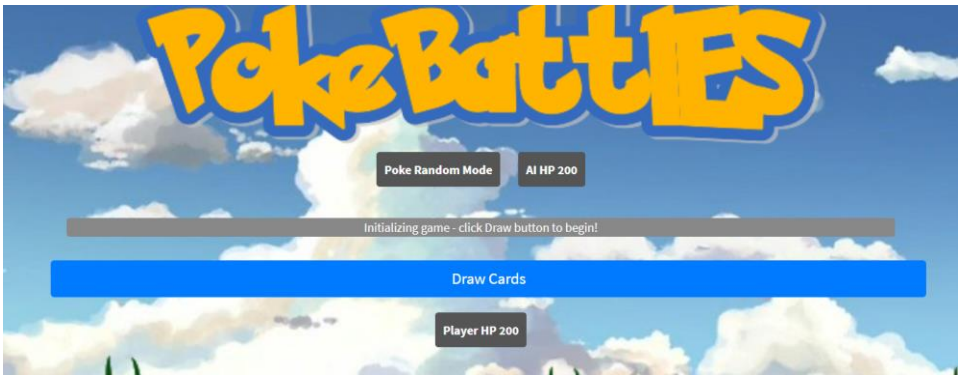
Game Setup Page

- This page sets up preliminary knowledge for our expert system. The user is able to set a custom HP, as well as a difficulty level for the AI. This page redirect user to Draw Page and starts keeping track of current state of knowledge.



Draw Page

- This page allows user to initialize the game or start a new turn. It contacts the pyknow project module to initialize a new turn and provides current user and AI HP, round details and other required features necessary for the Fact Engine to function.

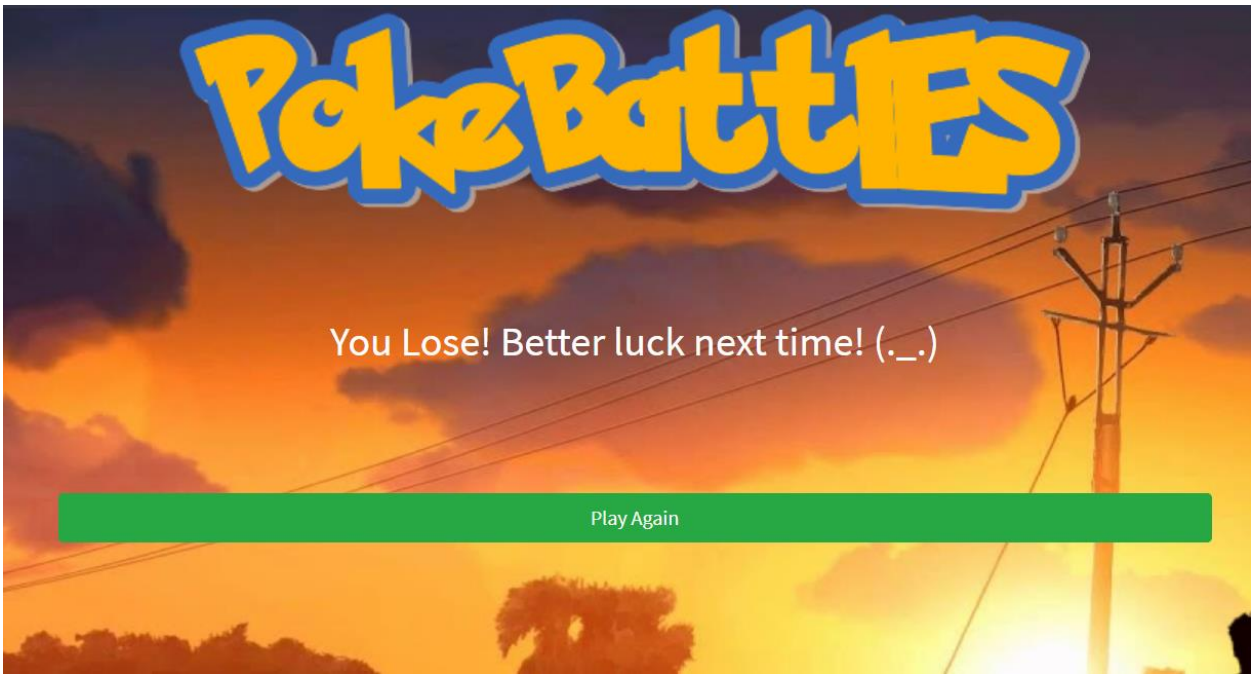


Play Page

- This page allows user to enter a selection of card which is then transferred to the rule-based expert system to run in addition to additional knowledge data, and to create the scenario based on user and AI selection. It redirects back to Draw or End page depending on current state.
- The results are returned in a string format back to the page and shown to user as detailed explanation of gameplay allowing them to visualize changes in HP easily.

End Game Page

- This page is the game finish page which is triggered when HP reaches 0 for either AI or the User. It allows the user to review the final result and also begin a new simulation, if desired.



The logic for the above pages is written using Flask App Routes and HTML Rendering Templates. The entire logic is available on GitHub for review [5].

Future Works

As future enhancements, the expert system can be further developed through:

- Addition of rules to manage gameplay scenarios.
- More minion types and rules to oversee battle logic for the respective types.
- Existing Rule-set updates to support a distinctive style of gameplay.
- Cross-platform support!

References

[1] https://en.wikipedia.org/wiki/Pok%C3%A9mon

[2] https://www.kaggle.com/abcsds/pokemon

[3] https://talons1337.github.io/pcm/

[4] https://github.com/buguroo/pyknow

[5] https://pokebattles.azurewebsites.net/

[6] https://github.com/tejas1794/PokeBattlES