Program code:

```
from flask import Flask, render_template, request, redirect, url_for, jsonify
import requests
import random
import math
import time
import json
import os

app = Flask(__name__)

# Cache for Pokémon data to reduce API calls
pokemon_cache = {}
move_cache = {}
@app.route('/', methods=['GET'])
def index():
    # Get list of Pokémon for the dropdown
    try:
        response = requests.get('https://pokeapi.co/api/v2/pokemon?limit=151')
        if response.status_code == 200:
            pokemon_data = []
            pokemon_dict = {}
            for pokemon in response.json()['results']:
                # Extract ID from URL for image
                pokemon_id = pokemon['url'].split('/')[-2]
                pokemon_name = pokemon['name']
```

```
            image_url = f"https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/{pokemon_id}.png"
            pokemon_data.append({
                'name': pokemon_name,
                'image': image_url
            })
            pokemon_dict[pokemon_name] = image_url

        pokemon_json = json.dumps(pokemon_dict)
        return render_template('index.html', pokemon_data=pokemon_data, pokemon_json=pokemon_json)
    else:
        pokemon_data = []
        pokemon_json = "{}"
except Exception as e:
    print(f"Error fetching Pokémon list: {e}")
    pokemon_data = []
    pokemon_json = "{}"

  return render_template('index.html', pokemon_data=pokemon_data, pokemon_json=pokemon_json)

@app.route('/battle', methods=['POST'])
def battle():
  user_pokemon = request.form.get('user_pokemon', '').lower().strip()
  opponent_pokemon = request.form.get('opponent_pokemon', '').lower().strip()
  user_trainer = request.form.get('user_trainer', '').lower().strip()
  opponent_trainer = request.form.get('opponent_trainer', '').lower().strip()
```

```
        if not all([user_pokemon, opponent_pokemon, user_trainer, opponent_trainer]):
        return render_template('index.html', error="Please select both trainers and Pokémon")
        # Fetch data for both Pokémon
    user_data = fetch_pokemon_data(user_pokemon)
    opponent_data = fetch_pokemon_data(opponent_pokemon)

    # Handle errors if either Pokémon wasn't found
    error_pokemon = None
    if not user_data:
        error_pokemon = user_pokemon
    elif not opponent_data:
        error_pokemon = opponent_pokemon



if error_pokemon:
        # Get fresh Pokémon data for the form
        try:
            response = requests.get('https://pokeapi.co/api/v2/pokemon?limit=151')
            pokemon_dict = {}
            for pokemon in response.json()['results']:
                pokemon_id = pokemon['url'].split('/')[-2]
                pokemon_name = pokemon['name']
                pokemon_dict[pokemon_name] =
f"https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/{pokemon_id}.png"


            pokemon_json = json.dumps(pokemon_dict)
```

```
            pokemon_data = [{'name': p, 'image': pokemon_dict[p]} for p in pokemon_dict]
            return render_template('index.html',
                                   pokemon_data=pokemon_data,
                                   pokemon_json=pokemon_json,
                                   error=f"Could not find Pokémon: {error_pokemon}")
        except Exception as e:
            print(f"Error preparing error response: {e}")
            return render_template('index.html', error=f"Could not find Pokémon:
{error_pokemon}")


    # Add trainer information
    user_data['trainer'] = user_trainer
    opponent_data['trainer'] = opponent_trainer


    return render_template('battle.html',



  user_pokemon=user_data,
                       opponent_pokemon=opponent_data)
@app.route('/attack', methods=['POST'])
def attack():
    try:
        data = request.json
        user_pokemon = data.get('user_pokemon')
        opponent_pokemon = data.get('opponent_pokemon')
        move_id = data.get('move_id')
```

```
    if not all([user_pokemon, opponent_pokemon, move_id]):
        return jsonify({'error': 'Missing required data'}), 400

    # Get move details
    move_data = fetch_move_data(move_id)

    # Calculate damage for user's attack
    damage = calculate_damage(user_pokemon, opponent_pokemon, move_data)

    # Update opponent's HP
    new_opponent_hp = max(0, opponent_pokemon['current_hp'] - damage)
    opponent_pokemon['current_hp'] = new_opponent_hp

    # Check if opponent fainted
    opponent_fainted = new_opponent_hp == 0
    damage': damage,
        'move_name': move_data['name'
'opponent_hp': new_opponent_hp,
        'opponent_hp_percent': (new_opponent_hp / opponent_pokemon['stats']['hp']) *
100,
        'opponent_fainted': opponent_fainted,
        'message': f"{user_pokemon['name']} used {move_data['name']}!"
    }
    # If opponent hasn't fainted, they counter-attack
    if not opponent_fainted:
        # Randomly select opponent's move
        opponent_move = random.choice(opponent_pokemon['moves'])
```

```
        opponent_move_data = fetch_move_data(opponent_move['id'])


        # Calculate damage for opponent's attack
        opponent_damage = calculate_damage(opponent_pokemon, user_pokemon,
opponent_move_data)


        # Update user's HP
        new_user_hp = max(0, user_pokemon['current_hp'] - opponent_damage)
        user_pokemon['current_hp'] = new_user_hp
        # Check if user fainted
        user_fainted = new_user_hp == 0
        result.update({
            'opponent_move': opponent_move_data['name'],
            'opponent_damage': opponent_damage,
            'user_hp': new_user_hp,
 'user_hp_percent': (new_user_hp / user_pokemon['stats']['hp']) * 100,
            'user_fainted': user_fainted,
            'opponent_message': f"{opponent_pokemon['name']} used
{opponent_move_data['name']}!"
        })
    return jsonify(result)
  except Exception as e:
    print(f"Error in attack route: {e}")
    return jsonify({'error': str(e)}), 500
def fetch_pokemon_data(pokemon_name):
  # Check cache first
  if pokemon_name in pokemon_cache:
```

```
    return pokemon_cache[pokemon_name]
try:
    response = requests.get(f'https://pokeapi.co/api/v2/pokemon/{pokemon_name}')
    if response.status_code == 200:
        data = response.json()
        # Extract the relevant information
        pokemon_data = {
            'name': data['name'].capitalize(),
            'image_front': data['sprites']['front_default'],
            'image_back': data['sprites']['back_default'],
            'stats': {},
            'types': [t['type']['name'] for t in data['types']],
   'moves': [],
            'current_hp': 0  # Will be set to max HP
        }
       # Extract stats
        for stat in data['stats']:
            stat_name = stat['stat']['name']
            stat_value = stat['base_stat']
            pokemon_data['stats'][stat_name] = stat_value

        # Ensure all required stats are present
        required_stats = ['hp', 'attack', 'defense', 'special-attack', 'special-defense', 'speed']
        for stat in required_stats:
            if stat not in pokemon_data['stats']:
                pokemon_data['stats'][stat] = 50  # Default value if missing
```

```
        # Set current HP to max HP
        pokemon_data['current_hp'] = pokemon_data['stats']['hp']

        # Get moves (limit to 4 for simplicity)
        moves_to_check = data['moves']
        if len(moves_to_check) > 10:
            moves_to_check = random.sample(data['moves'], 10)
        # Try to find damaging moves
        damaging_moves = []
        move_data in moves_to_check:
            move_url = move_data['move']['url']
            move_id = move_url.split('/')[-2]
             if move_id in move_cache:
                move_details = move_cache[move_id]
            else:
                move_response = requests.get(move_url)
                if move_response.status_code == 200:
                    move_details = move_response.json()
                    move_cache[move_id] = move_details
                else:
                    continue

        if move_details.get('power') is not None and move_details.get('damage_class',
{}).get('name') != 'status':
                move = {
                    'id': move_details['id'],
                    'name': move_details['name'].replace('-', ' ').title(),
```

```
                'type': move_details['type']['name'],
                'power': move_details['power'] or 0,
                'accuracy': move_details['accuracy'] or 100,
                'pp': move_details['pp']
            }
            damaging_moves.append(move)
            if len(damaging_moves) >= 4:
                break
if damaging_moves:
        pokemon_data['moves'] = damaging_moves
    else:
        pokemon_data['moves'] = [
            {
                'id': 1,
                'name': 'Tackle',
                'type': 'normal',
                'power': 40,
                'accuracy': 100,
                'pp': 35
            },
            {
                'id': 33,
                'name': 'Tackle',
                'type': 'normal',
                'power': 40,
                'accuracy': 100,
                'pp': 35
```

```
                }
            ]
        # Cache the Pokémon data
        pokemon_cache[pokemon_name] = pokemon_data
        return pokemon_data
    else:
  print(f"Error fetching {pokemon_name}: Status code {response.status_code}")
        return None
  except Exception as e:
    print(f"Error fetching {pokemon_name}: {e}")
    return None
def fetch_move_data(move_id):
  # Check cache first
  if move_id in move_cache:
    return move_cache[move_id]
  try:
    response = requests.get(f'https://pokeapi.co/api/v2/move/{move_id}')
    if response.status_code == 200:
      data = response.json()
      move_data = {
        'id': data['id'],
        'name': data['name'].replace('-', ' ').title(),
        'type': data['type']['name'],
        'power': data['power'] or 0,
        'accuracy': data['accuracy'] or 100,
        'pp': data['pp']
      }
```

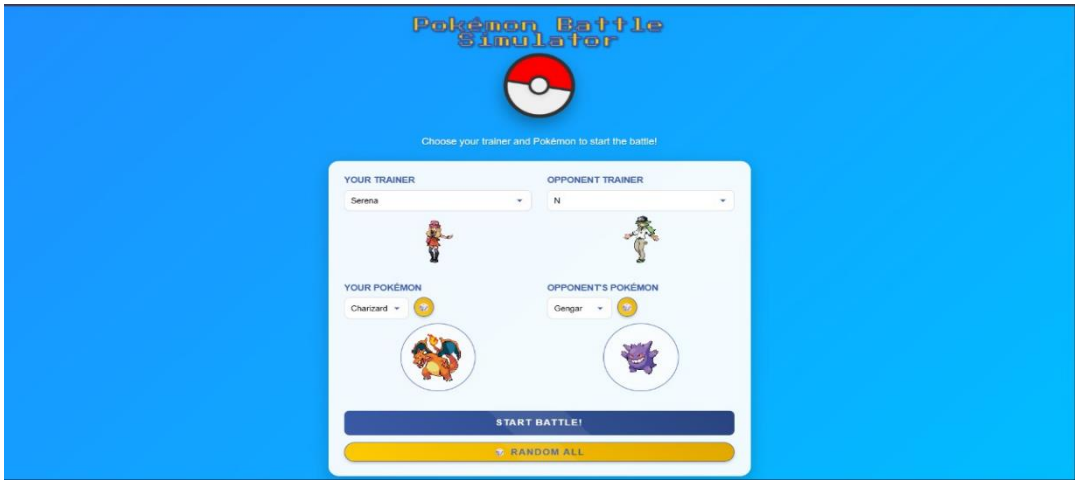
```
            # Cache the move data
            move_cache[move_id] = move_data
    return move_data
        else:
            # Return a default move if we can't fetch the data
            return {
                'id': 1,
                'name': 'Tackle',
                'type': 'normal',
                'power': 40,
                'accuracy': 100,
                'pp': 35
            }
    except Exception as e:
        print(f"Error fetching move {move_id}: {e}")
        # Return a default move if we can't fetch the data
        return {
            'id': 1,
            'name': 'Tackle',
            'type': 'normal',
            'power': 40,
            'accuracy': 100,
            'pp': 35
        }
def calculate_damage(attacker, defender, move):
    # Get the base power of the move
    power = move['power']
```

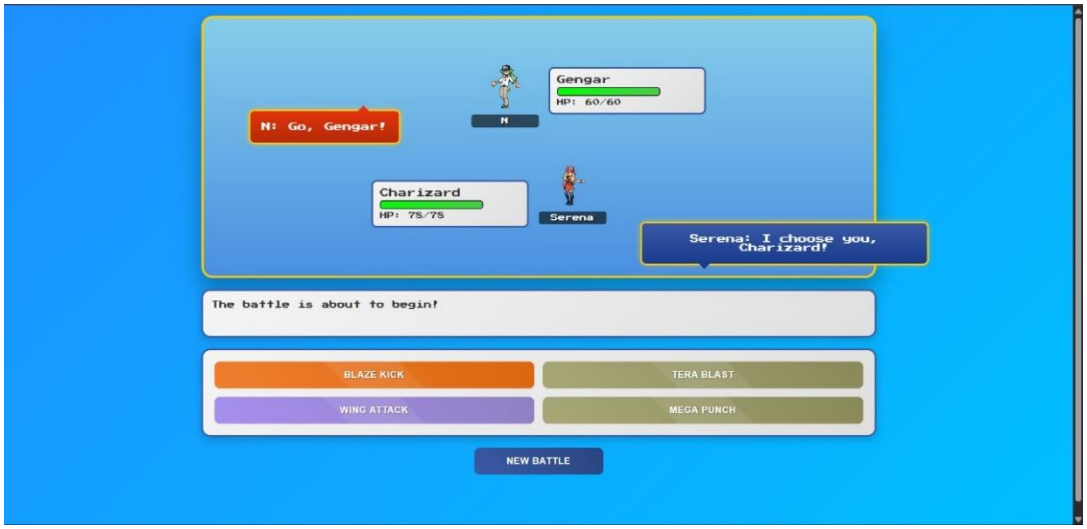
```
 # Get the attacker's relevant attack stat (physical or special)
    # In Gen 1-3, physical/special was determined by type
   physical_types = ['normal', 'fighting', 'flying', 'poison', 'ground', 'rock', 'bug', 'ghost', 'steel']


    if move['type'] in physical_types:
        attack_stat = attacker['stats']['attack']
        defense_stat = defender['stats']['defense']
    else:
        attack_stat = attacker['stats']['special-attack']
        defense_stat = defender['stats']['special-defense']
    # Calculate type effectiveness
    type_effectiveness = get_type_effectiveness(move['type'], defender['types'])
    # Calculate STAB (Same Type Attack Bonus)
    stab = 1.5 if move['type'] in attacker['types'] else 1.0
    # Calculate random factor (0.85 to 1.0)
    random_factor = random.uniform(0.85, 1.0)
    # Calculate damage using the Pokémon damage formula
    # Simplified version of the formula from Gen 3
    level = 50  # Assuming level 50 for simplicity
    damage = ((2 * level / 5 + 2) * power * attack_stat / defense_stat / 50 + 2) * stab *
type_effectiveness * random_factor
    return math.floor(damage)
def get_type_effectiveness(move_type,  defender_types):
    # Type effectiveness chart (simplified)
    type_chart = {
        'normal': {'rock': 0.5, 'ghost': 0, 'steel': 0.5},
```

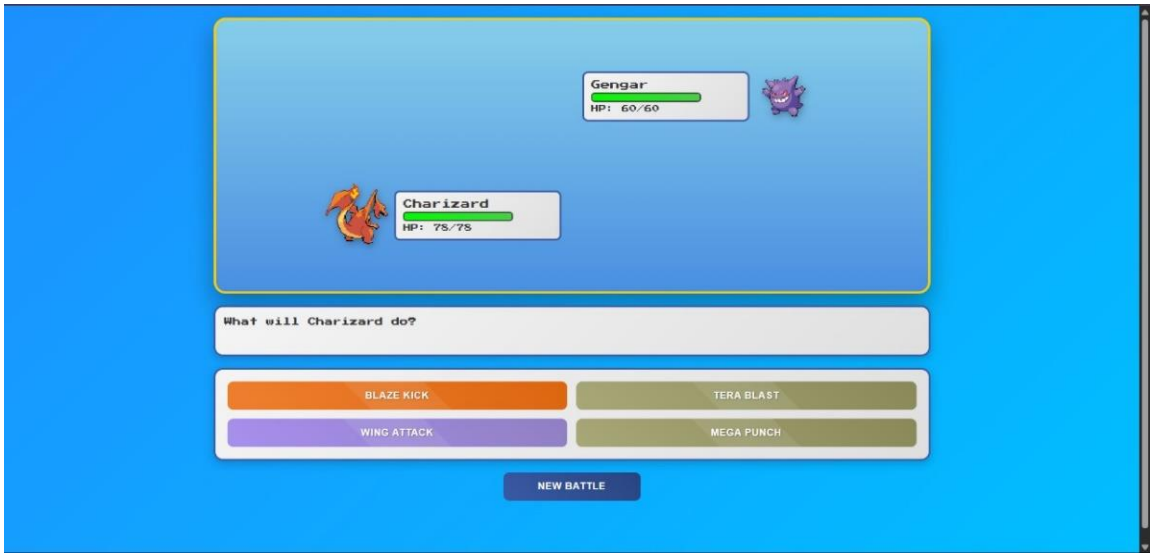
```
    'fire': {'fire': 0.5, 'water': 0.5, 'grass': 2, 'ice': 2, 'bug': 2, 'rock': 0.5, 'dragon': 0.5, 'steel': 2},
    'water': {'fire': 2, 'water': 0.5, 'grass': 0.5, 'ground': 2, 'rock': 2, 'dragon': 0.5},
    'electric': {'water': 2, 'electric': 0.5, 'grass': 0.5, 'ground': 0, 'flying': 2, 'dragon': 0.5},
    'grass': {'fire': 0.5, 'water': 2, 'grass': 0.5, 'poison': 0.5, 'ground': 2, 'flying': 0.5, 'bug': 0.5, 'rock': 2, 'dragon': 0.5, 'steel': 0.5},
    'ice': {'fire': 0.5, 'water': 0.5, 'grass': 2, 'ice': 0.5, 'ground': 2, 'flying': 2, 'dragon': 2, 'steel': 0.5},
    'fighting': {'normal': 2, 'ice': 2, 'poison': 0.5, 'flying': 0.5, 'psychic': 0.5, 'bug': 0.5, 'rock': 2, 'ghost': 0, 'dark': 2, 'steel': 2},
    'poison': {'grass': 2, 'poison': 0.5, 'ground': 0.5, 'rock': 0.5, 'ghost': 0.5, 'steel': 0},
    'ground': {'fire': 2, 'electric': 2, 'grass': 0.5, 'poison': 2, 'flying': 0, 'bug': 0.5, 'rock': 2, 'steel': 2},
    'flying': {'electric': 0.5, 'grass': 2, 'fighting': 2, 'bug': 2, 'rock': 0.5, 'steel': 0.5},
    'psychic': {'fighting': 2, 'poison': 2, 'psychic': 0.5, 'dark': 0, 'steel': 0.5},
    'bug': {'fire': 0.5, 'grass': 2, 'fighting': 0.5, 'poison': 0.5, 'flying': 0.5, 'psychic': 2, 'ghost': 0.5, 'dark': 2, 'steel': 0.5},
    'rock': {'fire': 2, 'ice': 2, 'fighting': 0.5, 'ground': 0.5, 'flying': 2, 'bug': 2, 'steel': 0.5},
    'ghost': {'normal': 0, 'psychic': 2, 'ghost': 2, 'dark': 0.5, 'steel': 0.5},
    'dragon': {'dragon': 2, 'steel': 0.5},
    'dark': {'fighting': 0.5, 'psychic': 2, 'ghost': 2, 'dark': 0.5, 'steel': 0.5},
    'steel': {'fire': 0.5, 'water': 0.5, 'electric': 0.5, 'ice': 2, 'rock': 2, 'steel': 0.5},
  }


  # Calculate effectiveness against multiple types
effectiveness = 1.0
  for defender_type in defender_types:
    if move_type in type_chart and defender_type in type_chart[move_type]:
      effectiveness *= type_chart[move_type][defender_type]
```

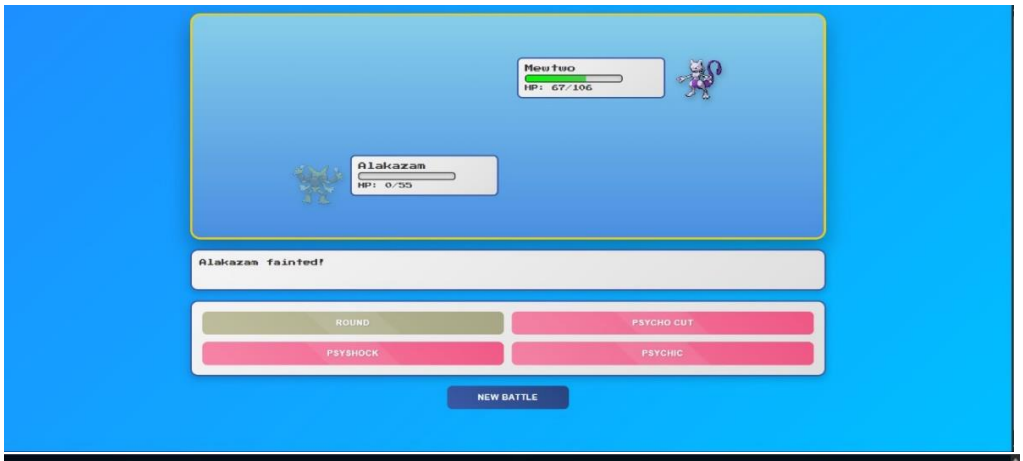
```
    return effectiveness

if __name__ == '__main__':
    port = int(os.environ.get('PORT', 5000))
    app.run(host='0.0.0.0', port=port, debug=True)
```

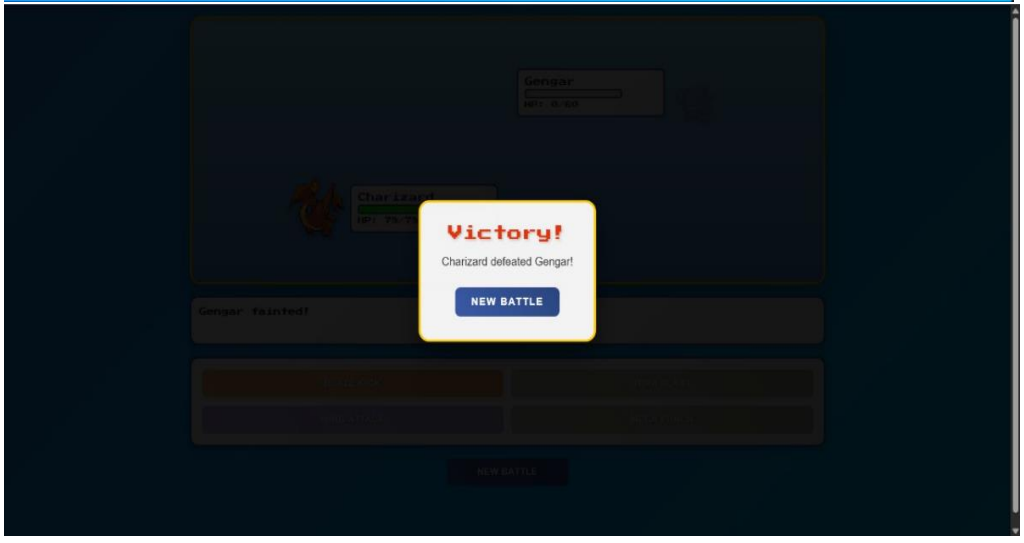
Output

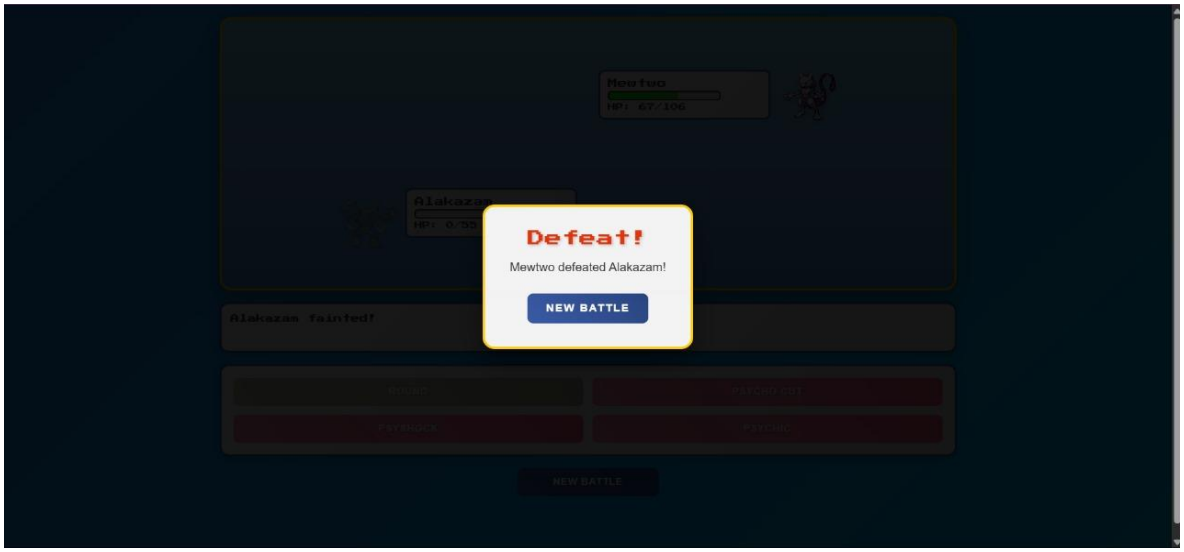





Gengar
HP: 60/60
Charizard
HP: 78/78
What will Charizard do?
BLAZE KICK
TERA BLAST
WING ATTACK
MEGA PUNCH
NEW BATTLE


Mewtwo
HP: 67/106
Alakazam
HP: 0/55
Alakazam fainted!
ROUND
PSYCHO CUT
PSYSHOCK
PSYCHIC
NEW BATTLE


Victory!
Charizard defeated Gengar!
NEW BATTLE



Conclusion

Hence we have implemented The Pokemon game in Python.