Group 10: VJ Davey, Duke Iko, Mackenzie Morrow Foster
CSCI 421 Database Systems

PokemonDB: Stage 4 + Appendices

The domain that we based our project off of is the pokemon domain. Pokemon are fictional creatures from a Gameboy game that are caught by Pokemon Trainers and used to battle each other. Many factors can influence a battle between two pokemon such as their types (water, fire, flying, grass, electric, etc.), their level (the more they fight and win, the higher their level), their attack moves, their defense moves, their speed, etc. Our application takes all of these factors into account and determines the overall weakness of a pokemon against certain types or the overall weakness of an entire team (up to 6 pokemon). After calculating the overall weaknesses, the information is displayed in a table on another page that lists all of the weaknesses of each pokemon against each type (e.g.Charmander (a fire type pokemon) is weak against water type pokemon).

All of the information and tables containing pokemon stats were obtained from a Github page populated with csv files regarding pokemon:
https://github.com/veekun/pokedex/tree/master/pokedex/data/csv.
We populated db with these tables and then instead of using psycopg to query our data, we used Django to make an object out of each table in the database and perform both raw queries and proxy queries with those objects instead. Using Python scripts, we wrote algorithms that could calculate the weaknesses and resistances that each pokemon had against each pokemon type. We then averaged the weaknesses and resistances of each pokemon to come up with the overall weakness of the team against each type and the overall resistance of the team against each type.

The SQL DDL Statements we used in producing this database included a CREATE statement for each table a la:

```
CREATE TABLE pokemon ( id integer, identifier varchar(25), species_id integer, height integer, weight integer, base_exp integer, num_order integer, is_default integer);
```

We would create a table which had a column for every column present in the CSV file we were drawing from. Following this, we populated each created table with information from the corresponding downloaded CSV files from the aforementioned github page a la:

```
\copy pokemon FROM '~akdavey/421/csv/pokemon.csv' DELIMITER ',' CSV;
```

After we had tables loaded with all the information contained in the CSV file, we thirdly would remove unused or useless information (in the context of our application) which was imported from the CSV by dropping columns:

```
ALTER TABLE pokemon DROP COLUMN num_order, DROP COLUMN is_default;
```

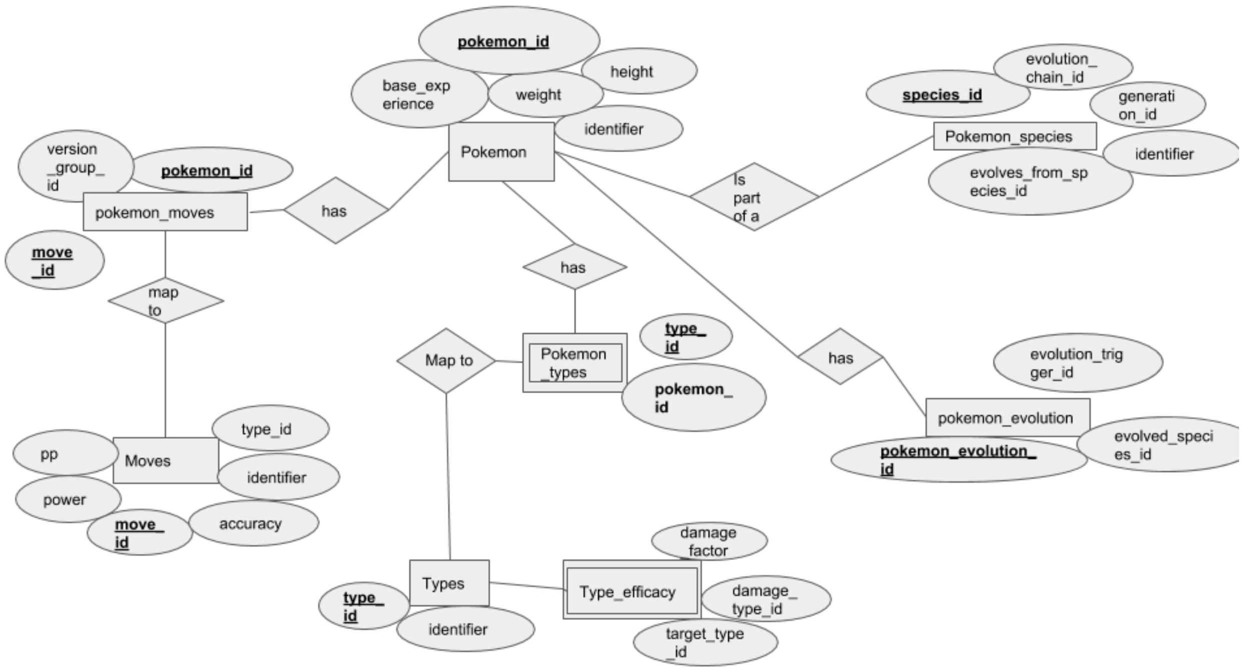
These steps were followed for every table in our database (sans Django generated tables used for managing migration history and admin/password information for our website) with the use of preexisting CSV files from the aforementioned github page. The CSV files on the page used to populate our database and power our application included:

- moves.csv
- pokemon.csv
- pokemon_evolution.csv
- pokemon_moves.csv
- pokemon_species.csv
- pokemon_types.csv
- type_efficacy.csv
- types.csv

As a final step for our DDL, we declare our primary keys in the Django application code and manage them from there in the myapp/models.py file.
Our resulting DDL is then as follows:

Pokemon(**<u>Pokemon_id</u>**, Height, Weight, Identifier, Base_experience)
Types(**<u>Type_id</u>**, Identifier)
Pokemon_types((**<u>Type_id, Pokemon_id</u>**))
  Foreign key Type_id references Types
  Foreign key pokemon_id reference Pokemon
Pokemon_species(**<u>Species_id</u>,** Evolution_chain_id, Generation_id, Identifier, Evolves_from_species_id)
Pokemon_moves((**<u>Pokemon_id</u>**, **Move_id**), Version_group_id)
  Foreign key Pokemon_id references Pokemon
  Foreign key Move_id references Moves
Pokemon_evolution(**<u>Pokemon_evolution_id</u>**, Evolution_trigger_id, Evolved_species_id)
Moves(**<u>Move_id,</u>** Type_id, PP, Power, Accuracy, Identifier)
Type_efficacy((**Target_type_id, Damage_type_id**), Damage_factor)
  Foreign key Target_type_id references Types
  Foreign key Damage_type_id references Types

The overall ER Diagram for our database is as follows:



Proof that our database is in BCNF is as follows:

Pokemon FD's:

Pokemon_id → Identifier, height, weight, base_experience

Therefore:

Pokemon_id → Pokemon_id, Identifier, height, weight, base_experience (via augmentation)

Pokemon_id is a superkey and all elements derived from it are subsets.

The schema is in BCNF

Pokemon_types FD:

*The foreign keys Pokemon_id and Slot, taken together act as the primary key.

(Pokemon_id, Slot)→Type_id

Therefore:

(Pokemon_id,Slot)→(Pokemon_id, Slot), Type_id

Since every pair of (Pokemon_id,Slot) uniquely determines a Type_id, this relation is in BCNF.

Pokemon_moves FD:

*The foreign keys Pokemon_id and Move_id , taken together, act as the primary key

(Pokemon_id, Move_id) → Version_group_id

Therefore:

(Pokemon_id, Move_id) → (Pokemon_id, Move_id), Version_group_id

Types FD:

Type_id → Identifier

Only two attributes in Types. The schema is in BCNF

Moves FD:

Move_id → Type_id, Identifier, Accuracy, Power, PP

Therefore:

Move_id → Move_id, Type_id, Identifier, Accuracy, Power, PP

Move_id is a superkey and all elements derived from it are subsets.

The schema is in BCNF

Pokemon_species FD:

Pokemon_species → Species_id, Identifier, Evolves_from_species_id, Generation_id, Evolution_chain_id

Therefore:

Pokemon_species → Pokemon_species, Species_id, Identifier, Evolves_from_species_id, Generation_id, Evolution_chain_id

Pokemon_species is a superkey and all elements derived from it are subsets.

The schema is in BCNF.

Pokemon_evolution FD:

Pokemon_evolution_id → Evolution_trigger_id, Evolved_species_id

Therefore:

Pokemon_evolution_id →Pokemon_evolution_id, Evolution_trigger_id, Evolved_species_id

Pokemon_evolution is a superkey and all elements derived from it are subsets.

The schema is in BCNF

Type_Efficacy FD:

*The foreign keys target_type_id and damage_type_id, used together, act as the primary keys of this schema

(target_type_id , damage_type_id) → damage_factor

Therefore:

(target_type_id , damage_type_id) → (target_type_id , damage_type_id) , damage_factor

After populating our database and doing setup with Django, we then used HTML to create several, simple model web pages to display the results of our queries and computations. We would use these simple model webpages with Django to dynamically create hundreds of webpages offering pokemon related information. Five model web pages were created under the ~templates folder:

- A main page where the user has the option to use the pokedex or our team builder
- A pokdex page that lists all 700+ pokemon with a clickable dropdown menu to each pokemon
- A dynamic details page that changes what information is displayed depending on which pokemon the user clicked from the page containing the list of pokemon
- A dynamic move page that lists the information of an attack move and all of the pokemon that can learn that move.
- A results page that displays the results of the queries and calculations to our team builder.

We used simple aspects of HTML to enhance the web pages to include things such as pictures, gifs, tables, colors, links, etc.

The main intention of the interface of our application is to take input from the user for 6 pokemon out of 700+ and return an analysis of their weaknesses, resistances, and immunities in a team environment. Given the name of the pokemon, the application queries the “pokemon”, “pokemon_types”, “type”, and “type_efficiacy” tables to gather, calculate, and display the information needed for the results web page. A minor functionality of our application is to provide a Pokedex (a device in the game and show that is a database containing information on all pokemon in the Pokemon universe) for the user to reference. The Pokedex page queries only the “pokemon” table since all that’s displayed is the pokemon’s name and ID. Then once a pokemon is selected, the application queries information from the “pokemon”, “pokemon_moves”, and “pokemon_species” tables to display all of the needed information for each pokemon’s individual page. Lastly, when a move is clicked on, the user is brought to a page that lists the information of the move and all of the pokemon that use that move which requires the application to query information from the “moves”, “pokemon_moves”, and “pokemon_species” tables.

The most notable aspect of our application is that the weaknesses, resistances, and immunities of each pokemon can be calculated, knowledge that can be very useful to anyone playing the Pokemon games who doesn’t like to lose. The player can use our website to determine the weaknesses and strengths of their pokemon against certain pokemon types which could significantly help them win a battle and level up, which is the main point of the game. In case the player encounters a pokemon in the game that they do not know, our application also functions as a Pokedex. Having information on all pokemon means that the database has information on 700+ pokemon, making the application up-to-date with the latest Pokemon game. So overall, this application gives important statistical information on the strength of a specific pokemon or a team of pokemon which will help its user determine the effectiveness of their team of pokemon in battle and potentially predict the outcome and determine how to best make adjustments to ensure victory.

APPENDIX A: Listing of Application Code

Our application is a Django Web application which dynamically generates web pages with the information in our database. On the department system, it is hosted inside of a folder named mysite on VJ's account in and has the following structure (Blue text indicates a directory):

```
|-mysite
|---myapp
|-----admin.py
|-----models.py
|-----tests.py
|-----urls.py
|-----views.py
|-----migrations
|-----static
|-------main_page
|-------pokdex_page
|-------pokemon_gifs
|-------results_page
|-------sugimori_art
|-----templates
|-------myapp
|---------detail.html
|---------index.html
|---------move_page.html
|---------pokedex.html
|---------results_page.html
|---mysite
|---manage.py
```

The root folder, mysite holds the important manage.py file, and the directories mysite and myapp. The mysite/mysite folder holds python files which are important to the inner workings of the Django site for administrative purposes. This is not our application code, but rather files generated during set up which help with maintaining the site. The manage.py file is also not our own application code, but is important to use to fire up the server.

The mysite/myapp folder is where our application resides. In the models.py, urls.py, and views.py files is code we created/generated ourselves which is used to dynamically generate webpages. The code for each of these files is formatted and replicated below:

models.py

```
# This is an auto-generated Django model module.
# You'll have to do the following manually to clean this up:
#   * Rearrange models' order
#   * Make sure each model has one field with primary_key=True
#   * Remove `managed = False` lines if you wish to allow Django to create and delete the table
# Feel free to rename the models, but don't rename db_table values or field names.
#
# Also note: You'll have to insert the output of 'django-admin.py sqlcustom [appname]'
# into your database.
from __future__ import unicode_literals

from django.db import models

class AuthGroup(models.Model):
    id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=80)
    class Meta:
        managed = False
        db_table = 'auth_group'

class AuthGroupPermissions(models.Model):
    id = models.IntegerField(primary_key=True)
    group = models.ForeignKey(AuthGroup)
    permission = models.ForeignKey('AuthPermission')
    class Meta:
        managed = False
        db_table = 'auth_group_permissions'

class AuthPermission(models.Model):
    id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=50)
    content_type = models.ForeignKey('DjangoContentType')
    codename = models.CharField(max_length=100)
    class Meta:
        managed = False
        db_table = 'auth_permission'

class AuthUser(models.Model):
    id = models.IntegerField(primary_key=True)
    password = models.CharField(max_length=128)
    last_login = models.DateTimeField()
    is_superuser = models.BooleanField()
    username = models.CharField(max_length=30)
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    email = models.CharField(max_length=75)
    is_staff = models.BooleanField()
    is_active = models.BooleanField()
    date_joined = models.DateTimeField()
    class Meta:
        managed = False
        db_table = 'auth_user'

class AuthUserGroups(models.Model):
    id = models.IntegerField(primary_key=True)
    user = models.ForeignKey(AuthUser)
    group = models.ForeignKey(AuthGroup)
    class Meta:
        managed = False
        db_table = 'auth_user_groups'

class AuthUserUserPermissions(models.Model):
    id = models.IntegerField(primary_key=True)
    user = models.ForeignKey(AuthUser)
    permission = models.ForeignKey(AuthPermission)
    class Meta:
        managed = False
        db_table = 'auth_user_user_permissions'

class DjangoAdminLog(models.Model):
    id = models.IntegerField(primary_key=True)
    action_time = models.DateTimeField()
    user = models.ForeignKey(AuthUser)
    content_type = models.ForeignKey('DjangoContentType', blank=True, null=True)
    object_id = models.TextField(blank=True)
    object_repr = models.CharField(max_length=200)
    action_flag = models.SmallIntegerField()
    change_message = models.TextField()
    class Meta:
        managed = False
        db_table = 'django_admin_log'

class DjangoContentType(models.Model):
    id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=100)
    app_label = models.CharField(max_length=100)
    model = models.CharField(max_length=100)
    class Meta:
        managed = False
        db_table = 'django_content_type'

class DjangoSession(models.Model):
    session_key = models.CharField(max_length=40)
    session_data = models.TextField()
    expire_date = models.DateTimeField()
    class Meta:
        managed = False
        db_table = 'django_session'

class Moves(models.Model):
    id = models.IntegerField(primary_key=True)
    identifier = models.CharField(max_length=25, blank=True)
    generation_id = models.IntegerField(blank=True, null=True)
    type_id = models.IntegerField(blank=True, null=True)
    power = models.IntegerField(blank=True, null=True)
    pp = models.IntegerField(blank=True, null=True)
    accuracy = models.IntegerField(blank=True, null=True)
    priority = models.IntegerField(blank=True, null=True)
    target_id = models.IntegerField(blank=True, null=True)
    damage_class_id = models.IntegerField(blank=True, null=True)
    effect_id = models.IntegerField(blank=True, null=True)
    effect_chance = models.IntegerField(blank=True, null=True)
    class Meta:
        managed = False
        db_table = 'moves'

class Pokemon(models.Model):
    id = models.IntegerField(primary_key=True)
    identifier = models.CharField(max_length=25, blank=True)
    species_id = models.IntegerField(blank=True, null=True)
    height = models.IntegerField(blank=True, null=True)
    weight = models.IntegerField(blank=True, null=True)
    base_exp = models.IntegerField(blank=True, null=True)

    def get_name(self):
            return self.identifier
    def get_id(self):
            return self.pokemon_id
    def get_species_id(self):
        return self.species_id
    def get_height(self):
            return self.height
    def get_weight(self):
            return self.weight
    def get_base_exp(self):
            return self.base_exp
    def __str__(self):
            return self.identifier

    class Meta:
        managed = False
        db_table = 'pokemon'


class PokemonEvolution(models.Model):
    id = models.IntegerField(primary_key=True)
    evolved_species_id = models.IntegerField(primary_key=True)
    evolution_trigger_id = models.IntegerField(blank=True, null=True)
    minimum_level = models.IntegerField(blank=True, null=True)
    class Meta:
        managed = False
        db_table = 'pokemon_evolution'

class PokemonMoves(models.Model):
    pokemon_id = models.IntegerField(primary_key=True)
    version_group_id = models.IntegerField(blank=True, null=True)
    move_id = models.IntegerField(primary_key=True)
    pokemon_move_method_id = models.IntegerField(blank=True, null=True)
    level = models.IntegerField(blank=True, null=True)
    class Meta:
        managed = False
        db_table = 'pokemon_moves'

class PokemonSpecies(models.Model):
    id = models.IntegerField(primary_key=True)
    identifier = models.CharField(max_length=25, blank=True)
    generation_id = models.IntegerField(blank=True, null=True)
    evolves_from_species_id = models.IntegerField(blank=True, null=True)
    evolution_chain_id = models.IntegerField(blank=True, null=True)
    class Meta:
        managed = False
        db_table = 'pokemon_species'

class PokemonTypes(models.Model):
    pokemon_id = models.IntegerField(primary_key=True)
    type_id = models.IntegerField(primary_key=True)
    slot = models.IntegerField(blank=True, null=True)

    def get_type_id(self):
      return self.type_id
    def get_pokemon_id(self):
      return self.pokemon_id
    def get_slot(self):
      return self.slot

    class Meta:
        managed = False
        db_table = 'pokemon_types'


class SouthMigrationhistory(models.Model):
    id = models.IntegerField(primary_key=True)
    app_name = models.CharField(max_length=255)
    migration = models.CharField(max_length=255)
    applied = models.DateTimeField()
    class Meta:
        managed = False
        db_table = 'south_migrationhistory'

class TypeEfficacy(models.Model):
    damage_type_id = models.IntegerField(blank=True, null=True)
    target_type_id = models.IntegerField(blank=True, null=True)
    damage_factor = models.IntegerField(blank=True, null=True)
    class Meta:
        managed = False
        db_table = 'type_efficacy'

class Types(models.Model):
    id = models.IntegerField(primary_key=True)
    identifier = models.CharField(max_length=20, blank=True)
    generation_id = models.IntegerField(blank=True, null=True)
    damage_class_id = models.IntegerField(blank=True, null=True)
```

```
class Meta:
    managed = False
    db_table = 'types'
```

urls.py

```
from django.conf.urls import url

from . import views

app_name = 'myapp'
urlpatterns = [
   # ex: /myapp/
    url(r'^$', views.index, name='index'),
   # ex: /myapp/pokedex
    url(r'^pokedex/$', views.pokedex, name='pokedex'),
   # ex: /myapp/pokedex/1
    url(r'^pokedex/(?P<pokemon_id>[0-9]+)/$', views.detail, name='detail'),
   # ex: myapp/results_page
    url(r'^results_page/$', views.results_page, name='results_page'),
   # ex: myapp/movedex/{moves.identifier}
    url(r'^movedex/(?P<move>[a-z\-]+)/$', views.move_page, name='move_page'),

]
```

views.py

```
from django.shortcuts import render, get_object_or_404
from django.template import loader
import math
from collections import OrderedDict
from django.db import connection
from operator import itemgetter
#databse stuff below
from .models import Pokemon, PokemonTypes, Types, Moves, PokemonMoves, PokemonSpecies
# Create your views here.

def type_efficacy_sql(user_list):
    #user list should be populated with either Pokemon Objects or 'None'
    cursor = connection.cursor()
    user_pkmn_type_efficacies=[]
    team_weaknesses={}
    team_resistances={}
    for item in user_list:
        if item != "None":
            pkmn_type_efficacy_dict={}
            #pull type_ids for pokemon
            pkmn_types = PokemonTypes.objects.filter(pokemon_id=item.id)
            for pkmn_type in pkmn_types:
                weakness_sql = "select distinct types.identifier from type_efficacy, types
where(type_efficacy.damage_type_id = types.id) and (type_efficacy.damage_factor > 100) and
type_efficacy.target_type_id=%d;"%(pkmn_type.type_id)
                cursor.execute(weakness_sql)
                weakness_fetch = cursor.fetchall()
                for item in weakness_fetch:
                    if item[0] not in pkmn_type_efficacy_dict:
                        pkmn_type_efficacy_dict.update({item[0]:2.0})

                    else:
                        current_value = pkmn_type_efficacy_dict.get(item[0])
                        pkmn_type_efficacy_dict.update({item[0]:(current_value *
2.0)})


                immunity_sql = "select distinct types.identifier as immunities from type_efficacy,
types where(type_efficacy.damage_type_id = types.id) and (type_efficacy.damage_factor = 0) and
type_efficacy.target_type_id=%d;"%(pkmn_type.type_id)
```

```
                                        cursor.execute(immunity_sql)
                                        immunity_fetch=cursor.fetchall()
                                        for item in immunity_fetch:
                                                if item[0] not in pkmn_type_efficacy_dict:
                                                        pkmn_type_efficacy_dict.update({item[0]:0})

                                                else:
                                                        pkmn_type_efficacy_dict.update({item[0]:0})



                                        resistance_sql="select distinct types.identifier as immunities from type_efficacy,
types where(type_efficacy.damage_type_id = types.id) and (type_efficacy.damage_factor < 100 and type_efficacy.damage_factor > 0)
and type_efficacy.target_type_id=%d;"%(pkmn_type.type_id)

                                        cursor.execute(resistance_sql)
                                        resistance_fetch=cursor.fetchall()
                                        for item in resistance_fetch:
                                                if item[0] not in pkmn_type_efficacy_dict:
                                                        pkmn_type_efficacy_dict.update({item[0]:0.5})

                                                else:
                                                        current_value = pkmn_type_efficacy_dict.get(item[0])
                                                        pkmn_type_efficacy_dict.update({item[0]:(current_value *
0.5)})



                                        normal_effectiveness_sql="select distinct types.identifier as immunities from
type_efficacy, types where(type_efficacy.damage_type_id = types.id) and (type_efficacy.damage_factor = 100) and
type_efficacy.target_type_id=%d;"%(pkmn_type.type_id)
                                        cursor.execute(normal_effectiveness_sql)
                        normal_fetch=cursor.fetchall()
                        for item in normal_fetch:
                            if item[0] not in pkmn_type_efficacy_dict:
                                pkmn_type_efficacy_dict.update({item[0]:1.0})

                            else:
                                current_value = pkmn_type_efficacy_dict.get(item[0])
                                pkmn_type_efficacy_dict.update({item[0]:(current_value * 1.0)})

                                 for key in pkmn_type_efficacy_dict:
                                        value = pkmn_type_efficacy_dict.get(key)
                                        if key not in team_weaknesses:
                                                if value > 1:
```

```
                        team_weaknesses.update({key:1})
                    if value <= 1:
                        team_weaknesses.update({key:0})
                else:
                    if value > 1:
                        currentval = team_weaknesses.get(key)
                        team_weaknesses.update({key:(currentval+1)})

                if key not in team_resistances:
                    if value < 1:
                        team_resistances.update({key:1})
                    if value >= 1:
                        team_resistances.update({key:0})
                else:
                    if value < 1:
                        currentval = team_resistances.get(key)
                        team_resistances.update({key:(currentval+1)})


            user_pkmn_type_efficacies.append(pkmn_type_efficacy_dict)
        else:
            user_pkmn_type_efficacies.append('None')

    print(team_resistances)
    print(team_weaknesses)
    user_pkmn_type_efficacies.append(team_weaknesses)
    user_pkmn_type_efficacies.append(team_resistances)
    return user_pkmn_type_efficacies





def index(request):
    pokemon_list = Pokemon.objects.order_by('id')
    context = []

    if request.GET:
        #time to do some magic
        name_inputs=[]
        user_list=[]
        for i in range(1,7):
            string = "pokemon%d"%(i)
```

```
                        if request.GET.get(string):
                            name_inputs.append(request.GET.get(string))
                        else:
                            name_inputs.append('None')
                    #name inputs holds all user inputs, for all user input we check if they are valid pokemon names
                    for name in name_inputs:
                        if name != 'None':
                            pokemon_exists = Pokemon.objects.filter(identifier=name)
                            if pokemon_exists:
                                user_list.append(Pokemon.objects.get(identifier=name))
                            else:
                                user_list.append("None")
                                print("User tried to add a pokemon named "+name+" to the team. No
pokemon exists with that name.")
                        else:
                            user_list.append("None")
                    type_efficacies = type_efficacy_sql(user_list)
                    #print(type_efficacies)
                    context = {'type_efficacies':type_efficacies, 'pokemonList':user_list}
                    return render(request, 'myapp/results_page.html',context)

                return render(request, 'myapp/index.html', context)

        def pokedex(request):
                pokemon_list = Pokemon.objects.order_by('id')
            context = {'pokemon_list': pokemon_list }
                return render(request, 'myapp/pokedex.html',context)

        def detail(request, pokemon_id):
                pokemon = get_object_or_404(Pokemon, id=pokemon_id)
                #pull pokemon types?
                pokemonTypeObject = PokemonTypes.objects.filter(pokemon_id=pokemon_id)
                typelist=[]
                for thing in pokemonTypeObject:
                    typeID = thing.get_type_id()
                    pokemonType = Types.objects.get(id=typeID)
                    typelist.append(pokemonType.identifier)
            #used to return the correct corner picture for pokemon that have official artwork available by Ken Sugimori
                number = '%0*d' % (3, int(pokemon_id))
                #math for calculating height in feet and inches
                heightininches = pokemon.height * 4
                feet = math.floor(heightininches/12)
                inches = heightininches-(feet*12)
        height="%d ft %d in" % (feet, inches)
    #correct weight to kg and then multiply for pounds. High level math right here.
        weight = (float(pokemon.weight) / 10)*2.2



        #equivalent to the evolution chain query in psql, used to get ids for pokemon current pokemon evolves to/from
    #evolves from
        evolves_from='None'
        evolves_to=[]
        if int(pokemon_id) <= 721:
                evolves_from_id = PokemonSpecies.objects.get(id=pokemon_id).evolves_from_species_id
                evolves_from = Pokemon.objects.filter(id=evolves_from_id)
                if evolves_from:
                        evolves_from=evolves_from[0]
                else:
                        evolves_from='None'
                evolves_to_ids=PokemonSpecies.objects.filter(evolves_from_species_id=pokemon_id)
                if evolves_to_ids:
                        for item in evolves_to_ids:
                                evolves_to.append(Pokemon.objects.get(id=item.id))
                else:
                        evolves_to='None'

        #move query dawg
        level_up_move_list = [] #will be a list of tuples
        tmhm_move_list = []
        pokemonMoveObjects = PokemonMoves.objects.filter(pokemon_id=pokemon_id)
    for thing in pokemonMoveObjects:
                #if thing.level > 0:
                        moveID = thing.move_id
                        move=Moves.objects.get(id=moveID)
                        moveTypeIdent=(Types.objects.get(id=move.type_id)).identifier
                        newitem=[int(thing.level),move.identifier, moveTypeIdent, move.power,
move.accuracy,move.pp]
                        if newitem not in level_up_move_list and (thing.version_group_id == 15 or
thing.version_group_id == 16):
                                if (thing.pokemon_move_method_id == 1):
                                        level_up_move_list.append(newitem)

                        if newitem not in tmhm_move_list and (thing.version_group_id == 15 or
thing.version_group_id == 16):
                                if (thing.pokemon_move_method_id == 4):
                                        tmhm_move_list.append(newitem[1:])
                        level_up_move_list.sort(key=lambda x: x[0])



                        return render(request, 'myapp/detail.html', {'pokemon':pokemon, 'types':typelist, 'number': number,
'height':height, 'weight':weight,'evolves_from':evolves_from, 'evolves_to': evolves_to, 'level_up_move_list':level_up_move_list ,
'tmhm_move_list':tmhm_move_list})


            def move_page(request, move):
                move_chosen = get_object_or_404(Moves, identifier=move)
                sql = "select id, identifier from pokemon_species where id IN (select distinct pokemon_moves.pokemon_id from
pokemon_moves, moves where (pokemon_moves.move_id = moves.id) and (moves.identifier = '%s'))"%(move);
                cursor = connection.cursor()
                cursor.execute(sql)
                pokemon_fetch = cursor.fetchall() #get a list of all the pokemon capable of learning this move by their ids and
identifiers
                #sql to fetch type of a move
                sql = "select types.identifier from types, moves where moves.type_id = types.id and moves.identifier='%s';"%(move)
                cursor.execute(sql)
                type_fetch = cursor.fetchall()[0][0]
                context={'move_chosen':move_chosen, 'pokemon_fetch':pokemon_fetch, "type":type_fetch}
                return render(request, 'myapp/move_page.html', context)


            def results_page(request):
                        context=None
                        #will need to adjust parameters here, want page to be passed full teams and calculations done with a python
script
                        return render(request, 'myapp/results_page.html', context)
```

The other files in this directory (mysite/myapp) are for administrative purposes.

Also included in this directory are the directories static, migrations, and templates. myapp/static contains media we use to bring glitter to our webpages. myapp/migrations (which was autogenerated and not explicity coded by us) contains code and information used with south to manage migrations between our database and our Django application. myapp/templates contains the subfolder myapp (a convention used to keep in line with the official Django tutorial) and several html files which are used as basic models for html webpages. These pages are populated with information from the database and the myapp/static folder to dynamically generate webpages on the fly. The HTML code is formatted and replicated below:

detail.html

```
<!DOCTYPE html>
<html>



<style>
h1 {
   color:black;
   font-family:georgia;
   font-size:300%;
   }

body {
                         background-image: url("../../../static/sugimori_art/{{number}}.png");
                         background-repeat: no-repeat;
            background-attachment: fixed;
                   background-position: right bottom;
                         background-size: 30% 30%;
             }
            img.resize{
                        height:15%;
                        width:10%;
            }

</style>
<p> Click <a href = "{% url 'myapp:pokedex' %}">here</a> to go back to the Main Pokedex page.</p>
<h1>{{ pokemon.identifier }}</h1>
```

```
        <img  src = "../../../static/pokemon_gifs/{{ pokemon.identifier }}.gif" alt="Sorry, we dont have an image for this pokemon
right now!" />

        <p>
        <b>Basic Information</b>
        <table border="5" style="background-color:gray">
         <tr>
           <td style="color:white"><b>Pokedex ID #:  {{ pokemon.id }}</b></td>
           <td></td>
         </tr>

         {% for type in types %}
         <tr>
           <td style="color:white"><b>Type {{forloop.counter }} : </b><img src="../../../static/results_page/{{type}}.jpg"></td>
           <td></td>
         </tr>
         {% endfor %}

         <tr>
           <td style="color:white"><b>Height : {{ height }}</b></td>
           <td></td>
         </tr>
         <tr>
           <td style="color:white"><b>Weight : {{ weight }} lbs</b></td>
           <td></td>
         </tr>
         <!--<tr>
           <td style="color:white"><b>Abilities</b></td>
           <td></td>
         </tr>-->
         {% if evolves_to != 'None' %}
         {% for evolution in evolves_to %}
           <tr>
             <td style="color:white"><b>Evolves to: <a href = "../{{evolution.id}}">{{evolution.identifier}}</a></b></td>
             <td></td>
           </tr>
         {% endfor %}
         {% endif %}
         {% if evolves_from != 'None' %}
         <tr>
           <td style="color:white"><b>Evolves from : <a href ="../{{evolves_from.id}}">{{evolves_from.identifier}}</a></b></td>
           <td></td>
         </tr>
```

```
  {% endif %}
</table>
</p>

<b>Moves Learned by Leveling Up</b>
<table border="5" style="background-color:gray">
  <tr>
    <th style="color:white">Level</th>
    <th style="color:white">Move Name</th>
    <th style="color:white">Type</th>
    <th style="color:white">Power</th>
    <th style="color:white">Accuracy</th>
    <th style="color:white">PP</th>
  </tr>
  {% for item in level_up_move_list %}
  <tr>
    {% for thing in item %}
    {% if forloop.counter = 3 %}
      <td><img src="../../../static/results_page/{{thing}}.jpg"></td>
     {% elif forloop.counter = 2 %}
              <td><a href = "../../movedex/{{thing}}">{{thing}}</a></td>
     {% else %}
      <td>{{thing}}</td>
     {% endif %}
    {% endfor %}
  </tr>
  {% endfor %}
</table>
</p>


<b>Moves Learned by TH/HM</b>
<table border="5" style="background-color:gray">
  <tr>
    <th style="color:white">Move Name</th>
    <th style="color:white">Type</th>
    <th style="color:white">Power</th>
    <th style="color:white">Accuracy</th>
    <th style="color:white">PP</th>
  </tr>
  {% for item in tmhm_move_list %}
  <tr>
    {% for thing in item %}
    {% if forloop.counter = 2 %}
       <td><img src="../../../static/results_page/{{thing}}.jpg"></td>
     {% elif forloop.counter = 1 %}
              <td><a href = "../../movedex/{{thing}}">{{thing}}</a></td>
     {% else %}
       <td>{{thing}}</td>
     {% endif %}
    {% endfor %}
  </tr>
  {% endfor %}
</table>
</p>
<pre>


</pre>





<!---<ul>



     <li>The base exp for this pokemon is : {{ pokemon.base_exp }}</li>
    {% for type in types %}
        <li>{{type}}</li>
    {% endfor %}
    <li>Height : {{ pokemon.weight }}</li>
    <li>Weight : {{ pokemon.height }}</li>

</ul>-->
```

index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"><html>
<head><META http-equiv="Content-Type" content="text/html; charset=utf-8">

<script type="text/javascript">
        function goToResults()
        {
          window.location = "{% url 'myapp:results_page' %}";
        }
</script>



</head>
<body>





<!--<div background="../../static/main_page/background.jpg">-->
<style>
body {
    background-image: url("../../static/main_page/background.jpg");
    background-attachment: fixed;
    background-size: 100% 100%;
}

</style>

<img src="../../static/main_page/ichooseyou.jpg" width="600" height="191">
<h1 style="font-size:200%">I Choose You! <img src="../../static/main_page/throwpokeballgif.gif" width="190"
height="130"></h1>

<hr>
<h2 style="font-family:impact">Pokedex</h2>
<p> Want to learn all about that pokemon you just saw? Click <a href= "{% url 'myapp:pokedex' %}">here!</a><p>

<h2 style="font-family:impact">Pokemon Calculator</h2>
<p>Don&#39;t understand why that other pokemon just kicked your butt? <br>
Don&#39;t understand how that adorable little animal just destroyed you?<br>
Just don&#39;t really understand in general why you lost?</p>
<p style="font-family:georgia"> <b>Well we can fix that! </b> <br>
Our Pokemon Calculator analyzes a team of pokemon and returns a list of all <br>
```

```
        their resistances and weaknesses that you can identify your own shortcomings or find out how to exploit your opponents
<br>
        (not so obvious) weaknesses. With this nifty tool, you can put together a team that&#39;s ready to</p>
        <p style="font-size:200%"> <b>WIN!</b></p>
        <img src="../../static/main_page/winning.gif" width="300" height="175">

        <hr>

        <h3 style="color:red">Pokemon Team Rater</h3>
        <p>List the 6 pokemon on your team below, then hit submit. <br>
        <form target="_blank" onsubmit="try {return window.confirm(&quot;You are submitting information to an external
page.\nAre you sure?&quot;);} catch (e) {return false;}">
        <fieldset>
          <legend>Enemy Team:</legend>
          Pokemon 1:<br>
          <input type="text" name="pokemon1"><br>
          Pokemon 2:<br>
          <input type="text" name="pokemon2"><br>
          Pokemon 3:<br>
          <input type="text" name="pokemon3"><br>
          Pokemon 4:<br>
          <input type="text" name="pokemon4"><br>
          Pokemon 5:<br>
          <input type="text" name="pokemon5"><br>
          Pokemon 6:<br>
          <input type="text" name="pokemon6"><br>
          <br>
          <input type="submit" value="Submit" onclick="goToResults()">
          </fieldset>
        </form>
        </p>

        <h3 style="color:blue">Your Team</h3>

        <p>Have fun picking and winning with your awesome new team!</p>
        <img src="../../static/main_page/poketeam.gif">

        </div>

        </body></html>
```

move_page.html

```
<html>
  <head>
    <title>{{ move_chosen.identifier }}</title>
    <meta content="">
    <style>
h1 {
    color:black;
    font-family:georgia;
    font-size:300%;
    }



</style>
  </head>
  <h1>{{ move_chosen.identifier }}</h1>

  <body>
  <p>Type : <img src="../../../static/results_page/{{type}}.jpg"> </p>
  <p>Power : {{ move_chosen.power }} </p>
  <p>Accuracy : {{ move_chosen.accuracy }}</p>
  <p>PP : {{ move_chosen.pp }}</p>

  <p>this move can be learned by the following pokemon:</p>
  <ul>
  {% for item in pokemon_fetch %}
  <li>
  {% for element in item %}
  {% if forloop.counter == 1 %}
  <a href = "../../pokedex/{{element}}">
  {% else %}
  {{ element }} </a>
  {% endif %}
  {% endfor %}
  </li>
  {% endfor %}
  </ul>

  <p> Click <a href = "{% url 'myapp:index' %}">here</a> to go back to the Main page.</p>
  </body>
</html>
```

pokedex.html

```
<html>
<head>
<script type="text/javascript">
        function goToNewPage()
        {
                var url = document.getElementById('list').value;
                if(url != 'none'){
                        window.location=url;
                }
        }
</script>
</head>
<body>

<style>

body{
        background-image: url("../../../static/pokdex_page/pokeball.png");
        background-attachment: fixed;
        background-size: 100% 100%;
}

</style>

        <p>Want to learn more about a pokemon? Select its name from the drop down list below. Alternatively, use the search box to
search for a pokemon's name to get its number at the end of the string. Then use the drop down list.</p><p> Click <a href = "{% url
'myapp:index' %}">here</a> to go back to the main page.</p>




{% if pokemon_list %}
  <!--<ul>
  {% for pokemon in pokemon_list %}
    <li><a href="{% url 'myapp:detail' pokemon.id %}">{{ pokemon.identifier }}</a></li>
  {% endfor %}
  </ul>-->
  <p> Drop Down Select:</p>
  <form name="pokedex"  method="get">
  <select name="pokedexdropdown" id="list" accesskey="target">
  <option value='none' selected>Choose a Pokemon</option>
```

```
   {% for pokemon in pokemon_list %}
            <option value="{% url 'myapp:detail' pokemon.id %}"># {{pokemon.id}} - {{pokemon.identifier}}</option>
   {% endfor %}
   </select>
   <input type=button value="Go" onclick="goToNewPage()" />
   </form>


  <p> Search Box:</p>
  <form action = "pokemonsearch" method="get">
  <input list="pokemon" name=pokemon">
  <datalist id="pokemon">
  {% for pokemon in pokemon_list %}
            <option value="{{pokemon.identifier}} - {{pokemon.id}}">
  {% endfor %}
  </datalist>

  </form>

{% else %}
   <p>No pokemon are available.</p>
{% endif %}
</body>
</html>
```

results_page.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"><html><head><META http-equiv="Content-Type"
content="text/html; charset=utf-8"></head><body>




<style>

body{
  background-image: url("../../../static/results_page/resultbackground.jpg");
  background-attachment: fixed;
  background-size: 100% 100%;
}

</style>
<h1 style="color:white">Your Team&#39;s Stats</h1>



<p>
<img src="../../../static/results_page/20years.jpg" style="float:right" width="725" height="300">
<table border="5">
  <tr style="background-color:black">
    <td></td>
    {% for item in pokemonList %}
    {% if item != 'None' %}
    <td style="color:white">{{ item.identifier }}</td>
    {% else %}
    <td style="color:white">None</td>
    {% endif %}
    {% endfor %}
    <td style="color:white"><b>Total Weak</b></td>
    <td style="color:white"><b>Total Resist</b></td>
  </tr>

  <tr style="background-color:lightgray">
    <td><img src="../../../static/results_page/normal.jpg"></td>
    {% for item in type_efficacies %}
      {% if item == 'None' %}
            <td>X</td>
      {% else %}
            {% if forloop.counter != 7 or forloop.counter != 8 %}
            {% for key, value in item.items %}
```

```
                {% if key == 'normal' %}
                  <td>{{ value }}</td>
                {% endif %}
              {% endfor %}
              {% elif forloop.counter == 7 %}
                {% for key, value in item.items %}
                  {% if key == 'normal' %}
                    <td style="background-color:red"></td>
                    {% endif %}
                {% endfor %}
              {% elif forloop.counter == 8 %}
                {% for key, value in item.items %}
                  {% if key == 'normal' %}
                    <td style="background-color:lime">{{ value }}</td>
                  {% endif %}
                {% endfor %}
              {% endif %}
    {% endif %}
  {% endfor %}
</tr>

<tr style="background-color:lightcoral">
  <td><img src="../../../static/results_page/fighting.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
              <td>X</td>
    {% else %}
              {% if forloop.counter != 7 or forloop.counter != 8 %}
              {% for key, value in item.items %}
                {% if key == 'fighting' %}
                  <td>{{ value }}</td>
                {% endif %}
              {% endfor %}
              {% elif forloop.counter == 7 %}
                {% for key, value in item.items %}
                  {% if key == 'fighting' %}
                    <td style="background-color:red"></td>
                    {% endif %}
                {% endfor %}
              {% elif forloop.counter == 8 %}
                {% for key, value in item.items %}
                  {% if key == 'fighting' %}
                    <td style="background-color:lime">{{ value }}</td>
                {% endif %}
              {% endfor %}
            {% endif %}
    {% endif %}
  {% endfor %}
</tr>



<tr style="background-color:lavender">
  <td><img src="../../../static/results_page/flying.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
            <td>X</td>
    {% else %}
            {% if forloop.counter != 7 or forloop.counter != 8 %}
            {% for key, value in item.items %}
              {% if key == 'flying' %}
                <td>{{ value }}</td>
              {% endif %}
            {% endfor %}
            {% elif forloop.counter == 7 %}
              {% for key, value in item.items %}
                {% if key == 'flying' %}
                  <td style="background-color:red"></td>
                  {% endif %}
              {% endfor %}
            {% elif forloop.counter == 8 %}
              {% for key, value in item.items %}
                {% if key == 'flying' %}
                  <td style="background-color:lime">{{ value }}</td>
                {% endif %}
              {% endfor %}
            {% endif %}
    {% endif %}
  {% endfor %}
</tr>


<tr style="background-color:mediumorchid">
  <td><img src="../../../static/results_page/poison.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
```

```
            <td>X</td>
    {% else %}
            {% if forloop.counter != 7 or forloop.counter != 8 %}
            {% for key, value in item.items %}
              {% if key == 'poison' %}
                <td>{{ value }}</td>
              {% endif %}
            {% endfor %}
            {% elif forloop.counter == 7 %}
              {% for key, value in item.items %}
                {% if key == 'poison' %}
                  <td style="background-color:red"></td>
                  {% endif %}
              {% endfor %}
            {% elif forloop.counter == 8 %}
              {% for key, value in item.items %}
                {% if key == 'poison' %}
                  <td style="background-color:lime">{{ value }}</td>
                {% endif %}
              {% endfor %}
            {% endif %}
    {% endif %}
  {% endfor %}
</tr>


<tr style="background-color:moccasin">
  <td><img src="../../../static/results_page/ground.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
            <td>X</td>
    {% else %}
            {% if forloop.counter != 7 or forloop.counter != 8 %}
            {% for key, value in item.items %}
              {% if key == 'ground' %}
                <td>{{ value }}</td>
              {% endif %}
            {% endfor %}
            {% elif forloop.counter == 7 %}
              {% for key, value in item.items %}
                {% if key == 'ground' %}
                  <td style="background-color:red"></td>
                  {% endif %}
              {% endfor %}
            {% elif forloop.counter == 8 %}
              {% for key, value in item.items %}
                {% if key == 'ground' %}
                  <td style="background-color:lime">{{ value }}</td>
                {% endif %}
              {% endfor %}
            {% endif %}
    {% endif %}
  {% endfor %}
</tr>


<tr style="background-color:tan">
  <td><img src="../../../static/results_page/rock.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
            <td>X</td>
    {% else %}
            {% if forloop.counter != 7 or forloop.counter != 8 %}
            {% for key, value in item.items %}
              {% if key == 'rock' %}
                <td>{{ value }}</td>
              {% endif %}
            {% endfor %}
            {% elif forloop.counter == 7 %}
              {% for key, value in item.items %}
                {% if key == 'rock' %}
                  <td style="backrock-color:red"></td>
                  {% endif %}
              {% endfor %}
            {% elif forloop.counter == 8 %}
              {% for key, value in item.items %}
                {% if key == 'rock' %}
                  <td style="backrock-color:lime">{{ value }}</td>
                {% endif %}
              {% endfor %}
            {% endif %}
    {% endif %}
  {% endfor %}
</tr>
```

```
<tr style="background-color:#b2c248">
  <td><img src="../../../static/results_page/bug.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
              <td>X</td>
    {% else %}
              {% if forloop.counter != 7 or forloop.counter != 8 %}
              {% for key, value in item.items %}
                {% if key == 'bug' %}
                  <td>{{ value }}</td>
                {% endif %}
              {% endfor %}
              {% elif forloop.counter == 7 %}
                {% for key, value in item.items %}
                  {% if key == 'bug' %}
                    <td style="backbug-color:red"></td>
                    {% endif %}
                {% endfor %}
              {% elif forloop.counter == 8 %}
                {% for key, value in item.items %}
                  {% if key == 'bug' %}
                    <td style="backbug-color:lime">{{ value }}</td>
                  {% endif %}
                {% endfor %}
              {% endif %}
    {% endif %}
  {% endfor %}
</tr>


<tr style="background-color:mediumpurple">
  <td><img src="../../../static/results_page/ghost.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
              <td>X</td>
    {% else %}
              {% if forloop.counter != 7 or forloop.counter != 8 %}
              {% for key, value in item.items %}
                {% if key == 'ghost' %}
                  <td>{{ value }}</td>
                {% endif %}
              {% endfor %}
              {% elif forloop.counter == 7 %}
```

```
        {% for key, value in item.items %}
          {% if key == 'ghost' %}
            <td style="backghost-color:red"></td>
            {% endif %}
        {% endfor %}
      {% elif forloop.counter == 8 %}
        {% for key, value in item.items %}
          {% if key == 'ghost' %}
            <td style="backghost-color:lime">{{ value }}</td>
          {% endif %}
        {% endfor %}
      {% endif %}
  {% endif %}
 {% endfor %}
</tr>


<tr style="background-color:ivory">
 <td><img src="../../../static/results_page/steel.jpg"></td>
 {% for item in type_efficacies %}
  {% if item == 'None' %}
      <td>X</td>
  {% else %}
      {% if forloop.counter != 7 or forloop.counter != 8 %}
      {% for key, value in item.items %}
        {% if key == 'steel' %}
          <td>{{ value }}</td>
        {% endif %}
      {% endfor %}
      {% elif forloop.counter == 7 %}
        {% for key, value in item.items %}
          {% if key == 'steel' %}
            <td style="backsteel-color:red"></td>
            {% endif %}
        {% endfor %}
      {% elif forloop.counter == 8 %}
        {% for key, value in item.items %}
          {% if key == 'steel' %}
            <td style="backsteel-color:lime">{{ value }}</td>
          {% endif %}
        {% endfor %}
      {% endif %}
  {% endif %}
  {% endfor %}
</tr>


<tr style="background-color:lightsalmon">
  <td><img src="../../../static/results_page/fire.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
		<td>X</td>
    {% else %}
		{% if forloop.counter != 7 or forloop.counter != 8 %}
		{% for key, value in item.items %}
		  {% if key == 'fire' %}
		    <td>{{ value }}</td>
		  {% endif %}
		{% endfor %}
		{% elif forloop.counter == 7 %}
		  {% for key, value in item.items %}
		    {% if key == 'fire' %}
		      <td style="backfire-color:red"></td>
		      {% endif %}
		  {% endfor %}
		{% elif forloop.counter == 8 %}
		  {% for key, value in item.items %}
		    {% if key == 'fire' %}
		      <td style="backfire-color:lime">{{ value }}</td>
		    {% endif %}
		  {% endfor %}
		{% endif %}
    {% endif %}
  {% endfor %}
</tr>

<tr style="background-color:lightblue">
  <td><img src="../../../static/results_page/water.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
		<td>X</td>
    {% else %}
		{% if forloop.counter != 7 or forloop.counter != 8 %}
		{% for key, value in item.items %}
		  {% if key == 'water' %}
		    <td>{{ value }}</td>
```

```
              {% endif %}
            {% endfor %}
            {% elif forloop.counter == 7 %}
              {% for key, value in item.items %}
                {% if key == 'water' %}
                  <td style="backwater-color:red"></td>
                  {% endif %}
              {% endfor %}
            {% elif forloop.counter == 8 %}
              {% for key, value in item.items %}
                {% if key == 'water' %}
                  <td style="backwater-color:lime">{{ value }}</td>
                {% endif %}
              {% endfor %}
            {% endif %}
    {% endif %}
  {% endfor %}
</tr>


<tr style="background-color:palegreen">
  <td><img src="../../../static/results_page/grass.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
            <td>X</td>
    {% else %}
            {% if forloop.counter != 7 or forloop.counter != 8 %}
            {% for key, value in item.items %}
              {% if key == 'grass' %}
                <td>{{ value }}</td>
              {% endif %}
            {% endfor %}
            {% elif forloop.counter == 7 %}
              {% for key, value in item.items %}
                {% if key == 'grass' %}
                  <td style="backgrass-color:red"></td>
                  {% endif %}
              {% endfor %}
            {% elif forloop.counter == 8 %}
              {% for key, value in item.items %}
                {% if key == 'grass' %}
                  <td style="backgrass-color:lime">{{ value }}</td>
                {% endif %}
              {% endfor %}
            {% endif %}
    {% endif %}
  {% endfor %}
</tr>


<tr style="background-color:khaki">
  <td><img src="../../../static/results_page/electric.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
            <td>X</td>
    {% else %}
            {% if forloop.counter != 7 or forloop.counter != 8 %}
            {% for key, value in item.items %}
              {% if key == 'electric' %}
                <td>{{ value }}</td>
              {% endif %}
            {% endfor %}
            {% elif forloop.counter == 7 %}
              {% for key, value in item.items %}
                {% if key == 'electric' %}
                  <td style="backelectric-color:red"></td>
                  {% endif %}
              {% endfor %}
            {% elif forloop.counter == 8 %}
              {% for key, value in item.items %}
                {% if key == 'electric' %}
                  <td style="backelectric-color:lime">{{ value }}</td>
                {% endif %}
              {% endfor %}
            {% endif %}
    {% endif %}
  {% endfor %}
</tr>


<tr style="background-color:lightpink">
  <td><img src="../../../static/results_page/psychic.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
            <td>X</td>
    {% else %}
```

```
              {% if forloop.counter != 7 or forloop.counter != 8 %}
              {% for key, value in item.items %}
                {% if key == 'psychic' %}
                  <td>{{ value }}</td>
                {% endif %}
              {% endfor %}
              {% elif forloop.counter == 7 %}
                {% for key, value in item.items %}
                  {% if key == 'psychic' %}
                    <td style="backpsychic-color:red"></td>
                    {% endif %}
                {% endfor %}
              {% elif forloop.counter == 8 %}
                {% for key, value in item.items %}
                  {% if key == 'psychic' %}
                    <td style="backpsychic-color:lime">{{ value }}</td>
                  {% endif %}
                {% endfor %}
              {% endif %}
    {% endif %}
  {% endfor %}
</tr>



<tr style="background-color:lightcyan">
  <td><img src="../../../static/results_page/ice.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
              <td>X</td>
    {% else %}
              {% if forloop.counter != 7 or forloop.counter != 8 %}
              {% for key, value in item.items %}
                {% if key == 'ice' %}
                  <td>{{ value }}</td>
                {% endif %}
              {% endfor %}
              {% elif forloop.counter == 7 %}
                {% for key, value in item.items %}
                  {% if key == 'ice' %}
                    <td style="backice-color:red"></td>
                    {% endif %}
                {% endfor %}
              {% elif forloop.counter == 8 %}
                {% for key, value in item.items %}
                  {% if key == 'ice' %}
                    <td style="backice-color:lime">{{ value }}</td>
                  {% endif %}
                {% endfor %}
              {% endif %}
    {% endif %}
  {% endfor %}
</tr>


<tr style="background-color:mediumslateblue">
  <td><img src="../../../static/results_page/dragon.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
              <td>X</td>
    {% else %}
              {% if forloop.counter != 7 or forloop.counter != 8 %}
              {% for key, value in item.items %}
                {% if key == 'dragon' %}
                  <td>{{ value }}</td>
                {% endif %}
              {% endfor %}
              {% elif forloop.counter == 7 %}
                {% for key, value in item.items %}
                  {% if key == 'dragon' %}
                    <td style="backdragon-color:red"></td>
                    {% endif %}
                {% endfor %}
              {% elif forloop.counter == 8 %}
                {% for key, value in item.items %}
                  {% if key == 'dragon' %}
                    <td style="backdragon-color:lime">{{ value }}</td>
                  {% endif %}
                {% endfor %}
              {% endif %}
    {% endif %}
  {% endfor %}
</tr>
<tr style="background-color:#6f4e37">
  <td><img src="../../../static/results_page/dark.jpg"></td>
  {% for item in type_efficacies %}
```

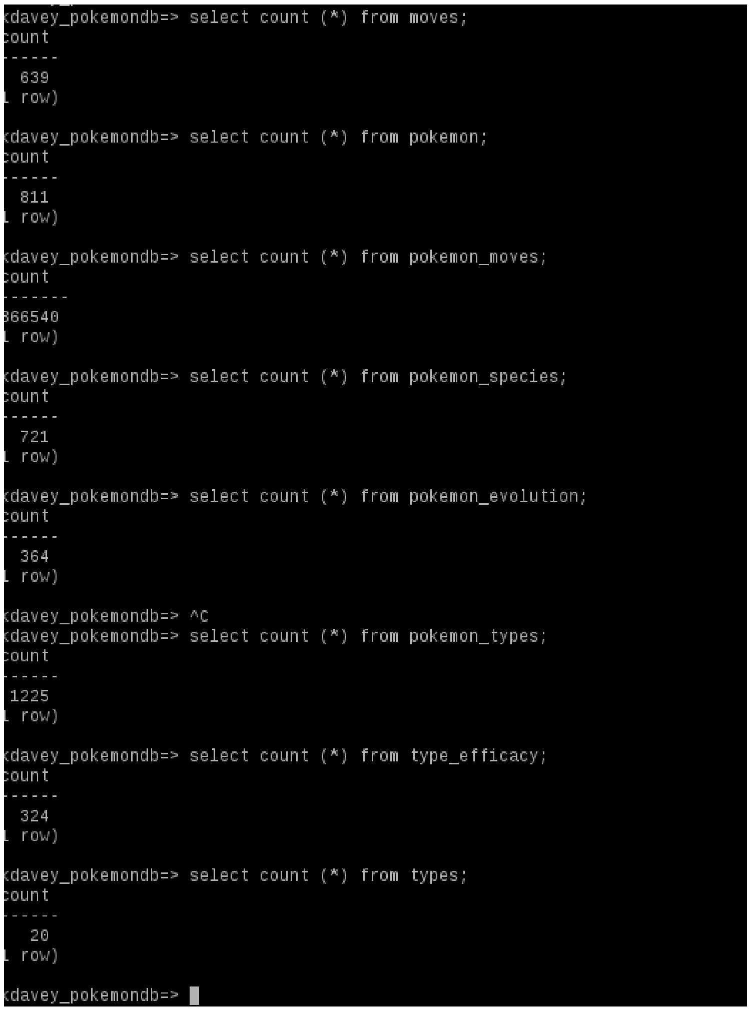
```
    {% if item == 'None' %}
              <td>X</td>
    {% else %}
              {% if forloop.counter != 7 or forloop.counter != 8 %}
              {% for key, value in item.items %}
                {% if key == 'dark' %}
                  <td>{{ value }}</td>
                {% endif %}
              {% endfor %}
              {% elif forloop.counter == 7 %}
                {% for key, value in item.items %}
                  {% if key == 'dark' %}
                    <td style="backdark-color:red"></td>
                    {% endif %}
                {% endfor %}
              {% elif forloop.counter == 8 %}
                {% for key, value in item.items %}
                  {% if key == 'dark' %}
                    <td style="backdark-color:lime">{{ value }}</td>
                  {% endif %}
                {% endfor %}
              {% endif %}
    {% endif %}
  {% endfor %}
</tr>


<tr style="background-color:plum">
  <td><img src="../../../static/results_page/fairy.jpg"></td>
  {% for item in type_efficacies %}
    {% if item == 'None' %}
              <td>X</td>
    {% else %}
              {% if forloop.counter != 7 or forloop.counter != 8 %}
              {% for key, value in item.items %}
                {% if key == 'fairy' %}
                  <td>{{ value }}</td>
                {% endif %}
              {% endfor %}
              {% elif forloop.counter == 7 %}
                {% for key, value in item.items %}
                  {% if key == 'fairy' %}
                    <td style="backfairy-color:red"></td>
```

```
                        {% endif %}
                      {% endfor %}
                    {% elif forloop.counter == 8 %}
                      {% for key, value in item.items %}
                        {% if key == 'fairy' %}
                          <td style="backfairy-color:lime">{{ value }}</td>
                        {% endif %}
                      {% endfor %}
                    {% endif %}
          {% endif %}
        {% endfor %}
      </tr>
    </table>
    </p>

    <p style="color:white">
    Tips to build a better team:</p>
    <ul style="color:white">
    <li>The numbers underneath your pokemon's name are the damage multipliers that occur when your pokemon is hit with an
attack of the type of that row.</li>
    <li>A multiplier of 1.0 means the attack will deal the normal amount of damage.</li>
    <li>A multiplier of 2.0 means that the attack will deal twice the amount of normal damage. Your pokemon will be weak to
attacks of this type</li>
    <li>A multiplier of 0.5 means that the attack will deal half the amount of normal damage. Your pokemon will be resistant to
attacks of this type</li>
    <li>A multiplier of 0 means that the attack will deal 0 damage. Your pokemon will be immune to attacks of this type</li>
    <li>Aim to have a Total Weak: Total Resist ratio where the number of Resistances is greater than the number of
weaknesses</li>
    <li>Compensate for one pokemon's weakness by including another pokemon that is resistant or immune to that
weakness</li>
    </ul>
    <p style="color:white">
    The more balanced your team is, the better you can handle anything anyone throws at you!
    </p>
    <p style="color:white">
    click <a href = "{% url 'myapp:index' %}">here</a>  to go back to the main page.
    </p>

    </div>

    </body></html>
```

This concludes the listing of our application code.

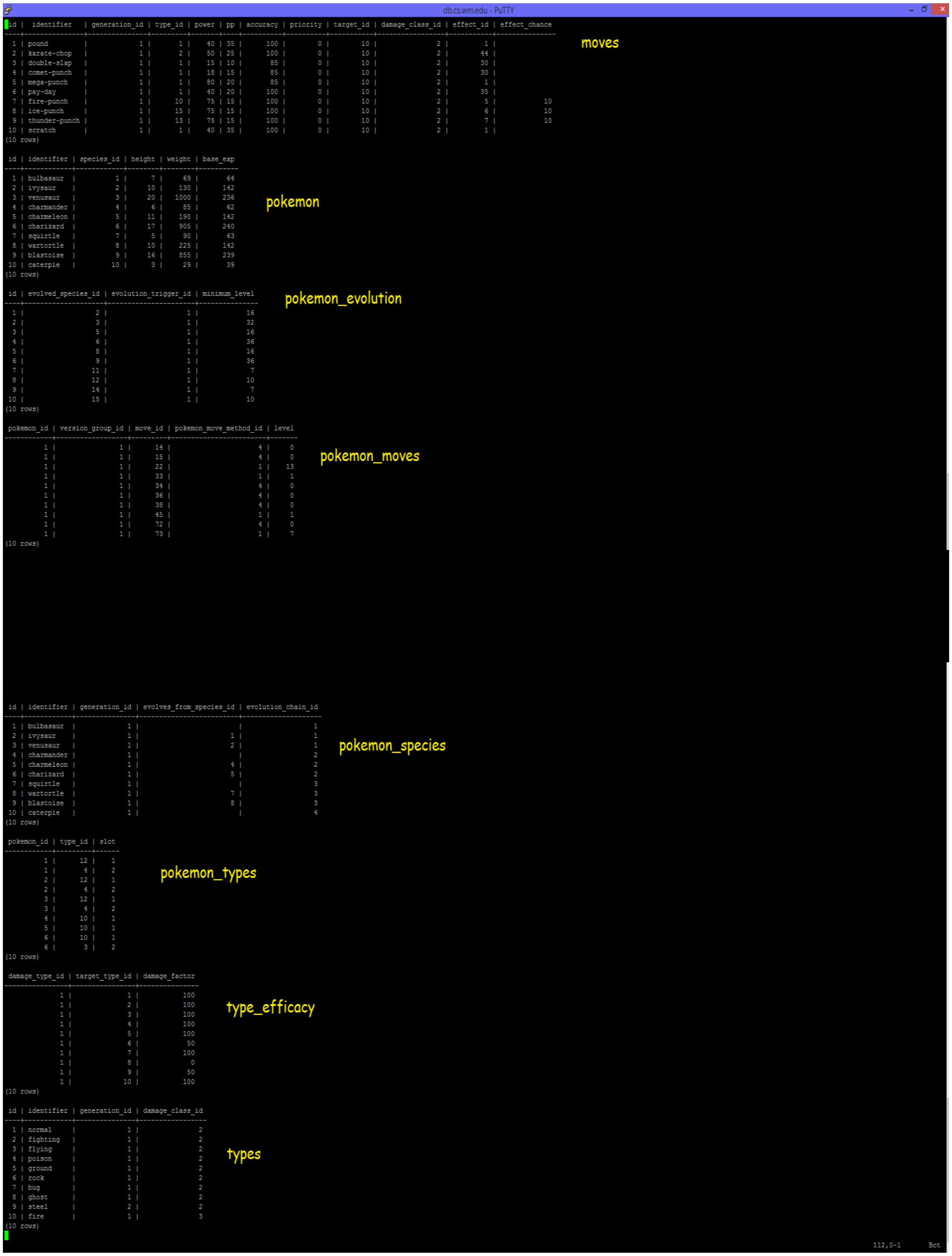
APPENDIX B: SELECT * FROM tablename LIMIT 10 on all tables in database. Also, SELECT COUNT(*) FROM tablename on all tables in database.

Select count(*):



```
kdavey_pokemondb=> select count (*) from moves;
count
------
  639
1 row)

kdavey_pokemondb=> select count (*) from pokemon;
count
------
  811
1 row)

kdavey_pokemondb=> select count (*) from pokemon_moves;
count
-------
366540
1 row)

kdavey_pokemondb=> select count (*) from pokemon_species;
count
------
  721
1 row)

kdavey_pokemondb=> select count (*) from pokemon_evolution;
count
------
  364
1 row)

kdavey_pokemondb=> ^C
kdavey_pokemondb=> select count (*) from pokemon_types;
count
------
 1225
1 row)

kdavey_pokemondb=> select count (*) from type_efficacy;
count
------
  324
1 row)

kdavey_pokemondb=> select count (*) from types;
count
------
   20
1 row)

kdavey_pokemondb=> 
```

Select * is listed on the next page.



```
 id |  identifier   | generation_id | type_id | power | pp | accuracy | priority | target_id | damage_class_id | effect_id | effect_chance
----+---------------+---------------+---------+-------+----+----------+----------+-----------+-----------------+-----------+---------------
  1 | pound         |             1 |       1 |    40 | 35 |      100 |        0 |        10 |               2 |         1 |
  2 | karate-chop   |             1 |       2 |    50 | 25 |      100 |        0 |        10 |               2 |        44 |
  3 | double-slap   |             1 |       1 |    15 | 10 |       85 |        0 |        10 |               2 |        30 |
  4 | comet-punch   |             1 |       1 |    18 | 15 |       85 |        0 |        10 |               2 |        30 |
  5 | mega-punch    |             1 |       1 |    80 | 20 |       85 |        0 |        10 |               2 |         1 |
  6 | pay-day       |             1 |       1 |    40 | 20 |      100 |        0 |        10 |               2 |        35 |
  7 | fire-punch    |             1 |      10 |    75 | 15 |      100 |        0 |        10 |               2 |         5 |            10
  8 | ice-punch     |             1 |      15 |    75 | 15 |      100 |        0 |        10 |               2 |         6 |            10
  9 | thunder-punch |             1 |      13 |    75 | 15 |      100 |        0 |        10 |               2 |         7 |            10
 10 | scratch       |             1 |       1 |    40 | 35 |      100 |        0 |        10 |               2 |         1 |
(10 rows)

 id | identifier | species_id | height | weight | base_exp
----+------------+------------+--------+--------+----------
  1 | bulbasaur  |          1 |      7 |     69 |       64
  2 | ivysaur    |          2 |     10 |    130 |      142
  3 | venusaur   |          3 |     20 |   1000 |      236
  4 | charmander |          4 |      6 |     85 |       62
  5 | charmeleon |          5 |     11 |    190 |      142
  6 | charizard  |          6 |     17 |    905 |      240
  7 | squirtle   |          7 |      5 |     90 |       63
  8 | wartortle  |          8 |     10 |    225 |      142
  9 | blastoise  |          9 |     16 |    855 |      239
 10 | caterpie   |         10 |      3 |     29 |       39
(10 rows)

 id | evolved_species_id | evolution_trigger_id | minimum_level
----+--------------------+----------------------+---------------
  1 |                  2 |                    1 |            16
  2 |                  3 |                    1 |            32
  3 |                  5 |                    1 |            16
  4 |                  6 |                    1 |            36
  5 |                  8 |                    1 |            16
  6 |                  9 |                    1 |            36
  7 |                 11 |                    1 |             7
  8 |                 12 |                    1 |            10
  9 |                 14 |                    1 |             7
 10 |                 15 |                    1 |            10
(10 rows)

 pokemon_id | version_group_id | move_id | pokemon_move_method_id | level
------------+------------------+---------+------------------------+-------
          1 |                1 |      14 |                      4 |     0
          1 |                1 |      15 |                      4 |     0
          1 |                1 |      22 |                      1 |    13
          1 |                1 |      33 |                      1 |     1
          1 |                1 |      34 |                      4 |     0
          1 |                1 |      36 |                      4 |     0
          1 |                1 |      38 |                      4 |     0
          1 |                1 |      45 |                      1 |     1
          1 |                1 |      72 |                      4 |     0
          1 |                1 |      73 |                      1 |     7
(10 rows)
```

```
 id | identifier | generation_id | evolves_from_species_id | evolution_chain_id
----+------------+---------------+-------------------------+--------------------
  1 | bulbasaur  |             1 |                         |                  1
  2 | ivysaur    |             1 |                       1 |                  1
  3 | venusaur   |             1 |                       2 |                  1
  4 | charmander |             1 |                         |                  2
  5 | charmeleon |             1 |                       4 |                  2
  6 | charizard  |             1 |                       5 |                  2
  7 | squirtle   |             1 |                         |                  3
  8 | wartortle  |             1 |                       7 |                  3
  9 | blastoise  |             1 |                       8 |                  3
 10 | caterpie   |             1 |                         |                  4
(10 rows)

 pokemon_id | type_id | slot
------------+---------+------
          1 |      12 |    1
          1 |       4 |    2
          2 |      12 |    1
          2 |       4 |    2
          3 |      12 |    1
          3 |       4 |    2
          4 |      10 |    1
          5 |      10 |    1
          6 |      10 |    1
          6 |       3 |    2
(10 rows)

 damage_type_id | target_type_id | damage_factor
----------------+----------------+---------------
              1 |              1 |           100
              1 |              2 |           100
              1 |              3 |           100
              1 |              4 |           100
              1 |              5 |           100
              1 |              6 |            50
              1 |              7 |           100
              1 |              8 |             0
              1 |              9 |            50
              1 |             10 |           100
(10 rows)

 id | identifier | generation_id | damage_class_id
----+------------+---------------+-----------------
  1 | normal     |             1 |               2
  2 | fighting   |             1 |               2
  3 | flying     |             1 |               2
  4 | poison     |             1 |               2
  5 | ground     |             1 |               2
  6 | rock       |             1 |               2
  7 | bug        |             1 |               2
  8 | ghost      |             1 |               2
  9 | steel      |             2 |               2
 10 | fire       |             1 |               3
(10 rows)
```

APPENDIX C: A list of all SQL statements that our application imposes upon the database server.

Python String Used to Query for all weaknesses (2x multipliers) for a given pokemon id:

"select distinct types.identifier from type_efficacy, types where(type_efficacy.damage_type_id = types.id) and (type_efficacy.damage_factor > 100) and type_efficacy.target_type_id=%d;"%(pkmn_type.type_id)

Python String Used to Query for all Immunities (0x multipliers) for a given pokemon id:

"select distinct types.identifier as immunities from type_efficacy, types where(type_efficacy.damage_type_id = types.id) and (type_efficacy.damage_factor = 0) and type_efficacy.target_type_id=%d;"%(pkmn_type.type_id)

Python String used to query for all resistances (.5x multipliers) for a given pokemon id:

"select distinct types.identifier from type_efficacy, types where(type_efficacy.damage_type_id = types.id) and (type_efficacy.damage_factor < 100 and type_efficacy.damage_factor > 0) and type_efficacy.target_type_id=%d;"%(pkmn_type.type_id)

Python String used to query for all normal effectiveness (1x multipliers) for a given pokemon id:

"select distinct types.identifier as immunities from type_efficacy, types where(type_efficacy.damage_type_id = types.id) and (type_efficacy.damage_factor = 100) and type_efficacy.target_type_id=%d;"%(pkmn_type.type_id)

Python String used to query for all Pokemon ids and identifiers that know a certain move:

"select id, identifier from pokemon_species where id IN (select distinct pokemon_moves.pokemon_id from pokemon_moves, moves where (pokemon_moves.move_id = moves.id) and (moves.identifier = '%s'))"%(move)

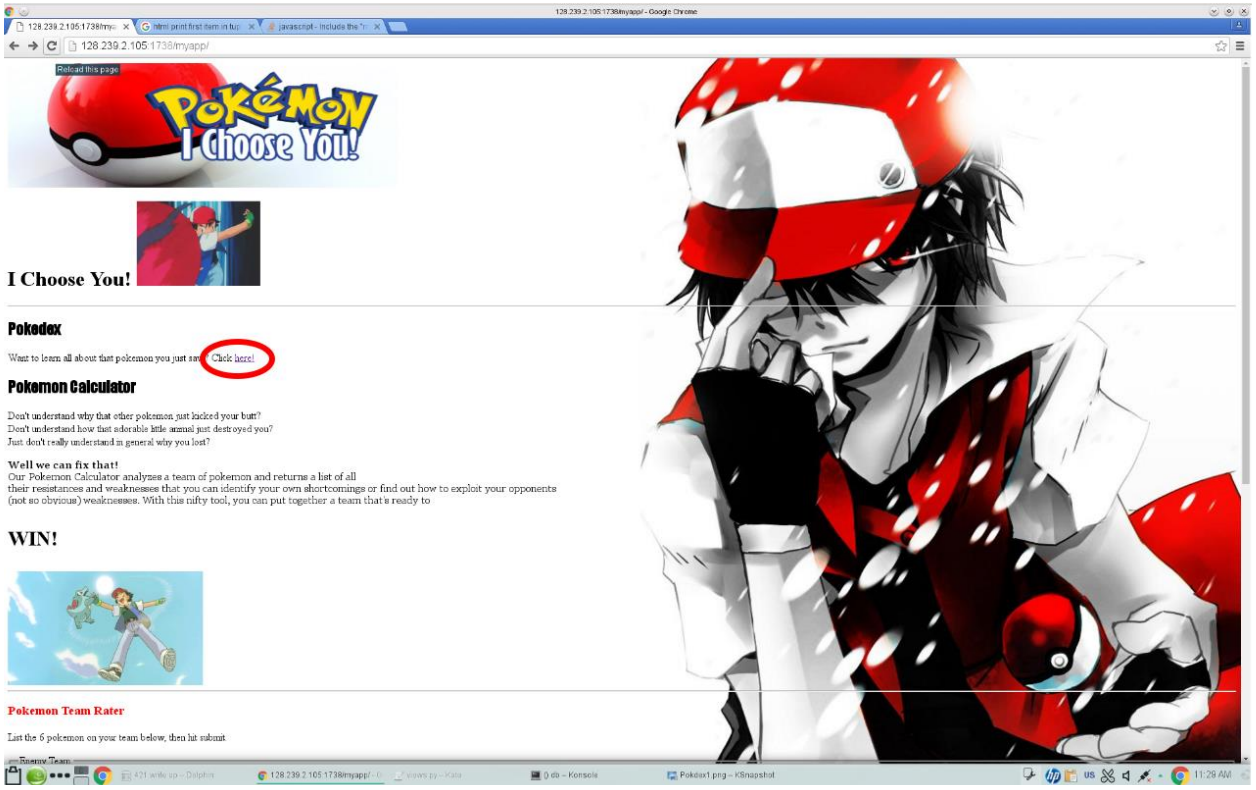
Python string used to query for the type of a move:

"select types.identifier from types, moves where moves.type_id = types.id and moves.identifier='%s';"%(move)

We additionally used Python/Django code to query the database via Django Model Objects before learning that we could do raw sql queries with the framework. We did this in the detail function in myapp/views.py. It should be readable and easy to tell how the database is queried by proxy.

APPENDIX D: Explicit Instructions on how to interact with our database using our application interface.

To begin using our application, you must first navigate to the root `mysite` directory in the shell. From there you must call "`python manage.py runserver 0.0.0.0:xxxx`", making sure to replace the x's with the name of a port on the machine you know you have access to. After this you should get a message which confirms the server is running. You can now open up a browser and go to the localhost and port address to access the website. Follow with "myapp" to specify that you wish to view our application, as shown in the picture below.


128.239.2.105:1738/myapp/ - Google Chrome
128.239.2.105:1738/mya
html print first item in tup
javascript - include the "
128.239.2.105:1738/myapp/
Reload this page
Pokémon
I Choose You!

I Choose You!

Pokedex

Want to learn all about that pokemon you just saw? Click here!

Pokemon Calculator

Don't understand why that other pokemon just kicked your butt?
Don't understand how that adorable little animal just destroyed you?
Just don't really understand in general why you lost?

Well we can fix that!
Our Pokemon Calculator analyzes a team of pokemon and returns a list of all
their resistances and weaknesses that you can identify your own shortcomings or find out how to exploit your opponents
(not so obvious) weaknesses. With this nifty tool, you can put together a team that's ready to

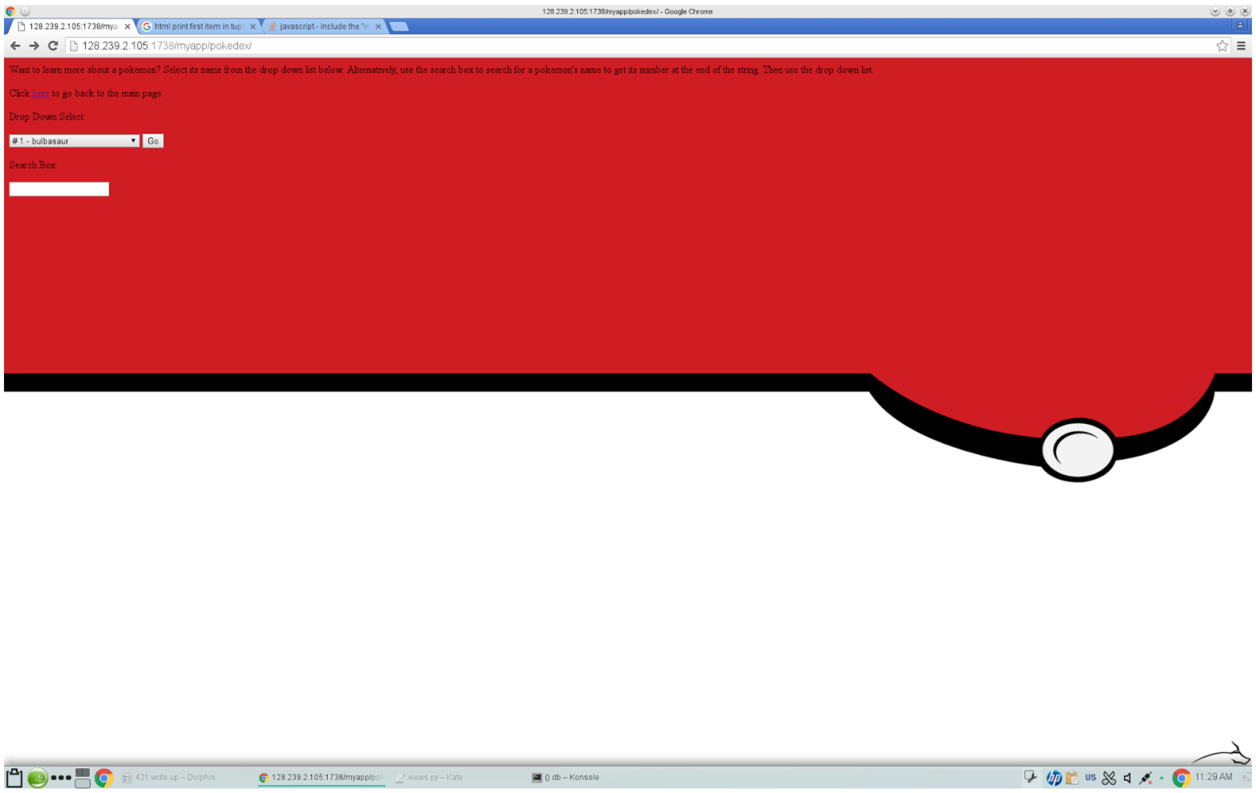
WIN!

Pokemon Team Rater

List the 6 pokemon on your team below, then hit submit

Enemy Team

Above is a screenshot of the main page of our website. There are two interactions that can be performed on the main page. The first of the two is clicking on the link (circled in red). That link will take you to another page that (shown below) that lists the names and IDs of all of the pokemon in the database.


128.239.2.105:1738/myapp/pokedex/ - Google Chrome
128.239.2.105:1738/mya
html print first item in tup
javascript - Include the "n
128.239.2.105:1738/myapp/pokedex/

Want to learn more about a pokemon? Select its name from the drop down list below. Alternatively, use the search box to search for a pokemon's name to get its number at the end of the string. Then use the drop down list.

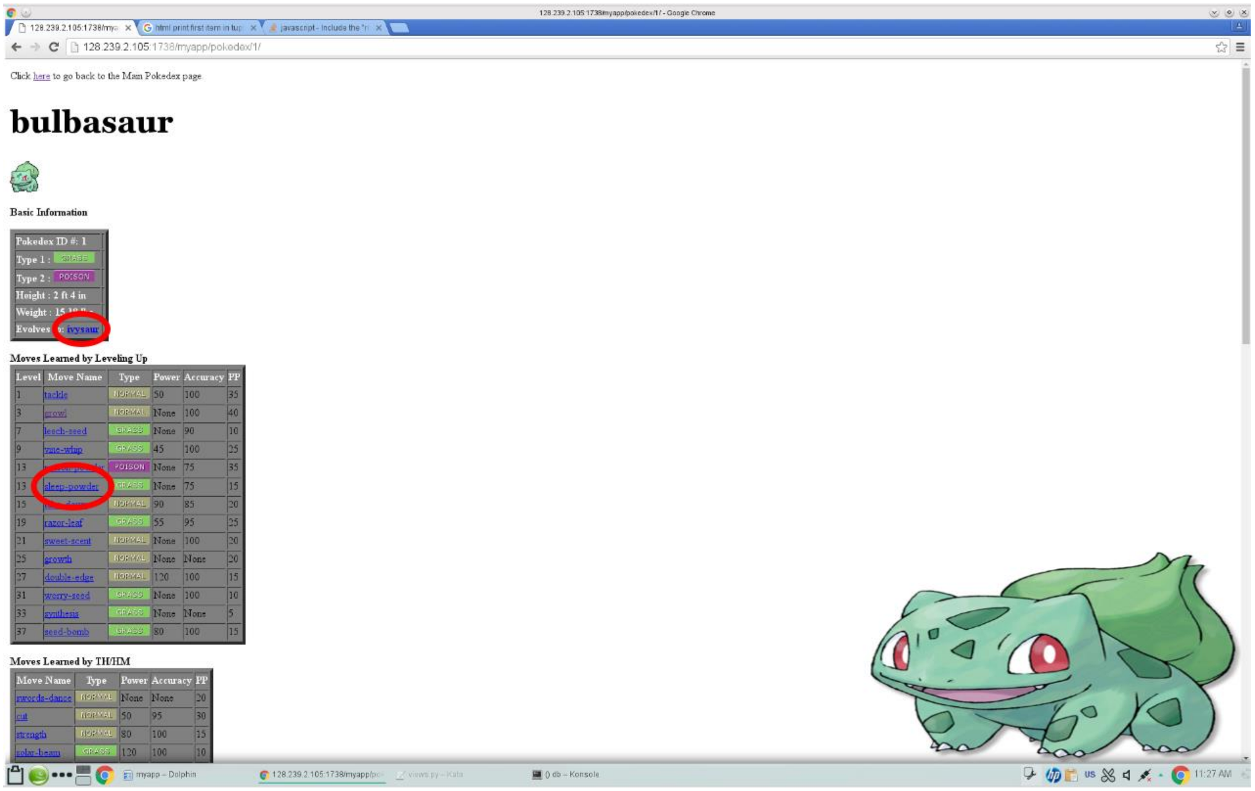
Click here to go back to the main page.

Drop Down Select:

Search Box:

421 write up – Dolphin
128.239.2.105:1738/myapp/po
views.py – Kate
0 db – Konsole
11:29 AM

This page is our "Pokedex" (a device in the game that contains information on all of the existing pokemon). You simply have to select a pokemon name from the drop down menu or enter the name into the "Search Box" and click the "Go" button where you will be taken to a page (shown below) that displays all of the basic information of that pokemon as well as its attack moves and TM&HM moves.



128.239.2.105:1738/myapp/pokedex/1/

Click here to go back to the Main Pokedex page

bulbasaur

Basic Information

Pokedex ID #: 1
Type 1 : GRASS
Type 2 : POISON
Height : 2 ft 4 in
Weight : 15.[illegible]
Evolves [illegible]: ivysaur

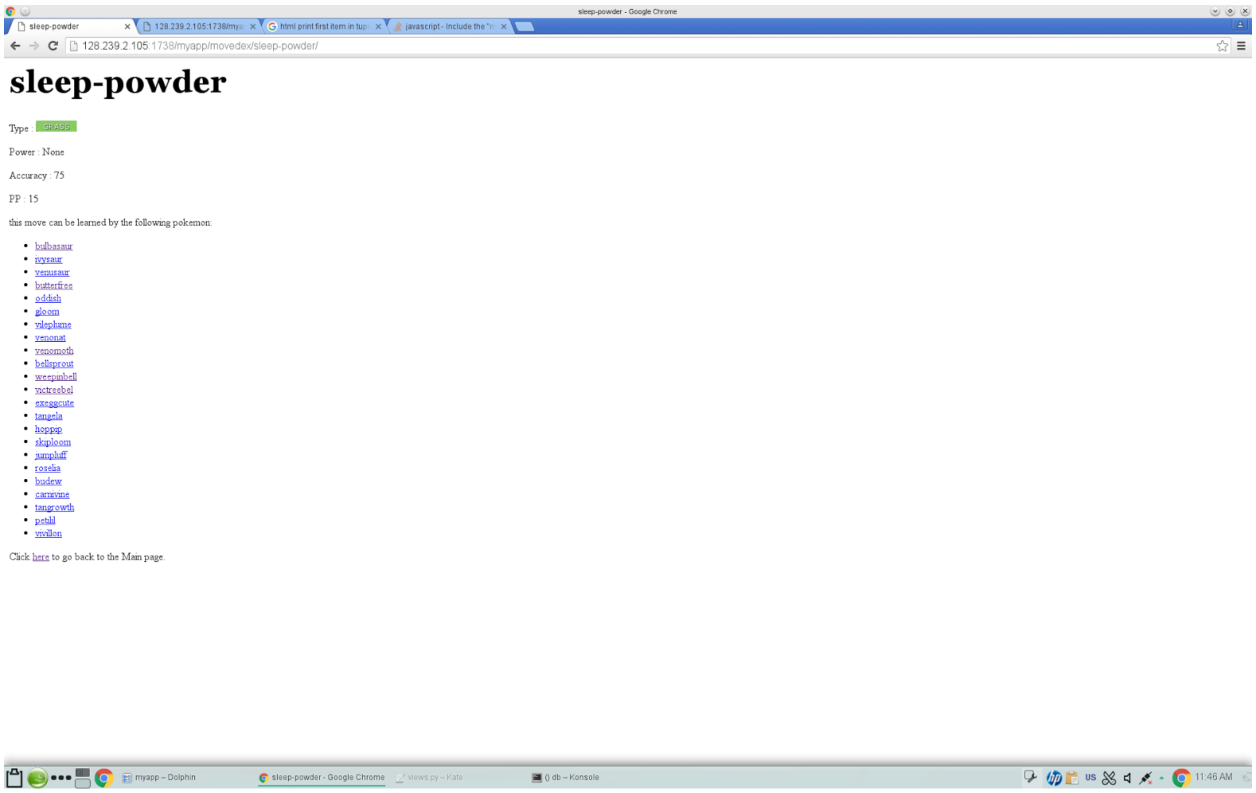
Moves Learned by Leveling Up

Level	Move Name	Type	Power	Accuracy	PP
1	tackle	NORMAL	50	100	35
3	growl	NORMAL	None	100	40
7	leech-seed	GRASS	None	90	10
9	vine-whip	GRASS	45	100	25
13	[illegible]	POISON	None	75	35
13	sleep-powder	GRASS	None	75	15
15	[illegible]	NORMAL	90	85	20
19	razor-leaf	GRASS	55	95	25
21	sweet-scent	NORMAL	None	100	20
25	growth	NORMAL	None	None	20
27	double-edge	NORMAL	120	100	15
31	worry-seed	GRASS	None	100	10
33	synthesis	GRASS	None	None	5
37	seed-bomb	GRASS	80	100	15

Moves Learned by TH/HM

Move Name	Type	Power	Accuracy	PP
swords-dance	NORMAL	None	None	20
cut	NORMAL	50	95	30
strength	NORMAL	80	100	15
solar-beam	GRASS	120	100	10

Shown here is an example Pokedex page for Bulbasaur. Listed is the pokemon's basic information: name, picture, types, height, weight, and evolution. Clicking on the evolution of the pokemon (in this case, "Ivysaur") will take you to the Pokedex page of that evolved pokemon (Bulbasaur evolves into Ivysaur which then evolves into Venusaur). Listed below the basic information are all of the attack moves that the pokemon can learn by leveling up. Clicking on one of the moves (in this case, "sleep-powder") will take you to a page that lists the name, type, accuracy, and PP of the move as well as all of the pokemon that can learn that move (shown below).


sleep-powder - Google Chrome
128.239.2.105:1738/myapp/movedex/sleep-powder/

sleep-powder

Type : GRASS

Power : None

Accuracy : 75

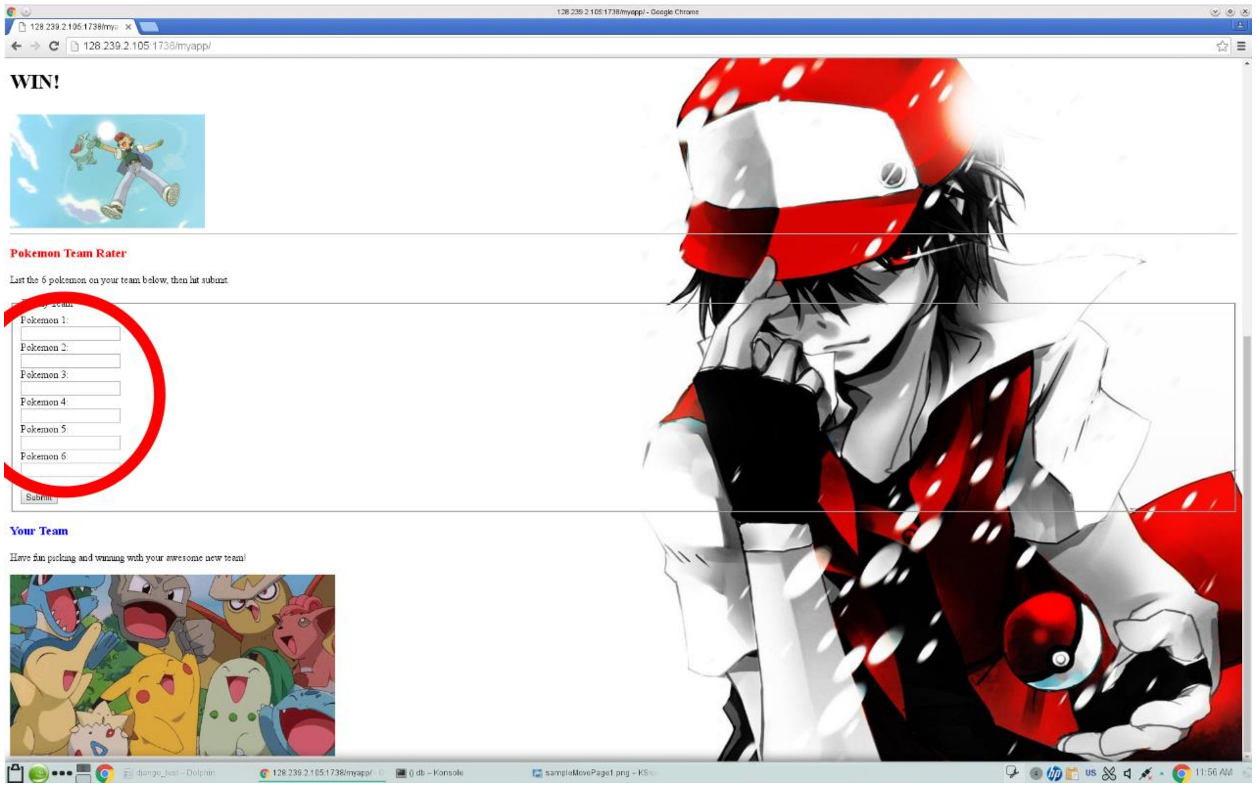
PP : 15

this move can be learned by the following pokemon:

- bulbasaur
- ivysaur
- venusaur
- butterfree
- oddish
- gloom
- vileplume
- venonat
- venomoth
- bellsprout
- weepinbell
- victreebel
- exeggcute
- tangela
- hoppip
- skiploom
- jumpluff
- roselia
- budew
- carnivine
- tangrowth
- petilil
- vivillon

Click here to go back to the Main page.

Shown here is the page that shows a moves, its information, and a list of the pokemon that can learn the move. Below the list is a link to go back to the main page of the website.


128.239.2.105:1738/myapp/ - Google Chrome
128.239.2.105:1738/myapp/

WIN!

Pokemon Team Rater

List the 6 pokemon on your team below, then hit submit.

Pokemon 1:

Pokemon 2:

Pokemon 3:

Pokemon 4:

Pokemon 5:

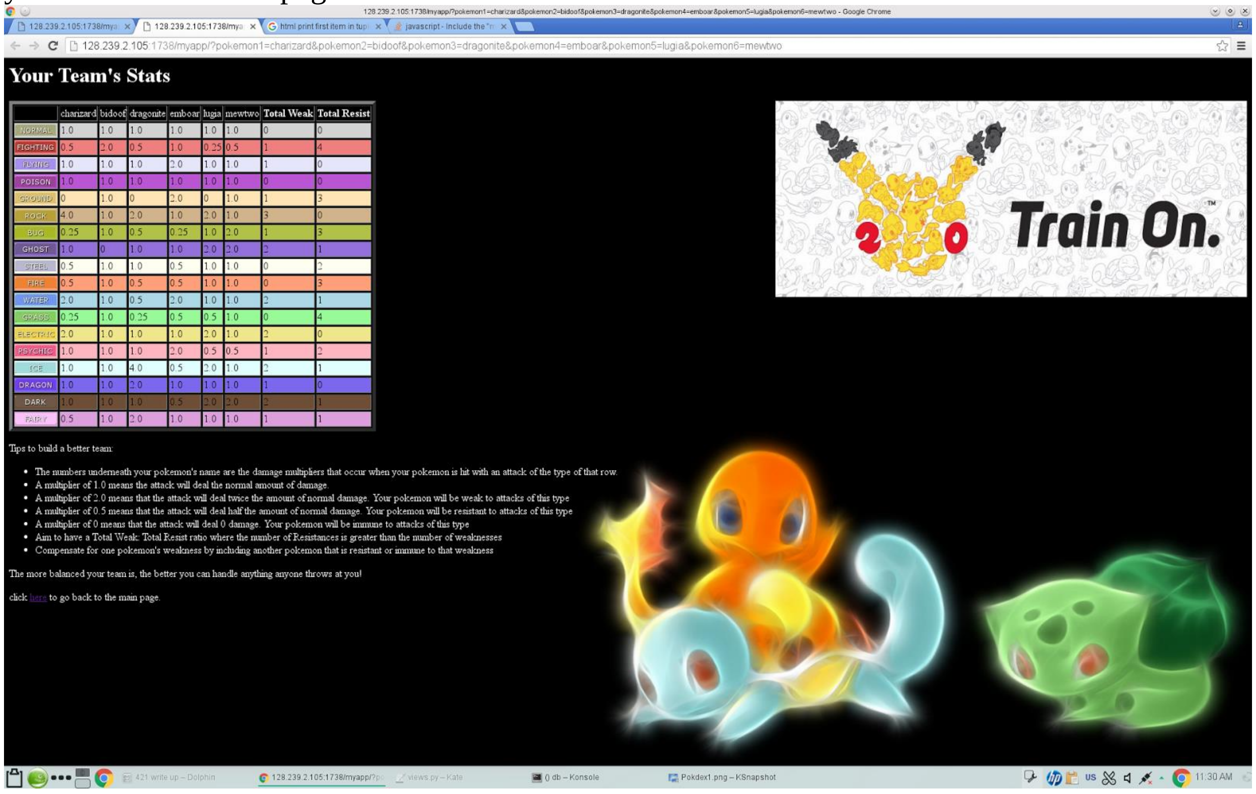
Pokemon 6:

Submit

Your Team

Have fun picking and winning with your awesome new team!

Back on the main page at the bottom of the page is the main functionality of our application. There are 6 input slots where you can input anywhere from 1-6 pokemon names to add to your team (note: typing in a name that is not registered as a pokemon name will result in a “none” type and will show a message in the terminal that the user tried to input a name that does not match any pokemon in the database. Upon clicking the “Submit” button, you will be taken to another page that lists all of the weaknesses of the inputted pokemon (shown below).
Shown here is the resulting page from the inputted pokemon on the main page. The names of the pokemon are listed in the table headers and below them are that pokemon’s weakness to each pokemon type (ex: Charizard, a fire type pokemon, is weak against water type pokemon). Listed below the table are the explanations for interpreting the results of the table. Any number greater than 1 means that the pokemon is is weak against that pokemon type. Any number less than 1 means that the pokemon has a resistance against that pokemon type; 0 means the pokemon is immune against that pokemon type. And a number 1 means that pokemon receives a normal amount of damage from that pokemon type. Displayed below the explanations is a link that takes you back to the main page.



Your Team's Stats

	charizard	bidoof	dragonite	emboar	lugia	mewtwo	Total Weak	Total Resist
NORMAL	1.0	1.0	1.0	1.0	1.0	1.0	0	0
FIGHTING	0.5	2.0	0.5	1.0	0.25	0.5	1	4
FLYING	1.0	1.0	1.0	2.0	1.0	1.0	1	0
POISON	1.0	1.0	1.0	1.0	1.0	1.0	0	0
GROUND	0	1.0	0	2.0	0	1.0	1	3
ROCK	4.0	1.0	2.0	1.0	2.0	1.0	3	0
BUG	0.25	1.0	0.5	0.25	1.0	2.0	1	3
GHOST	1.0	0	1.0	1.0	2.0	2.0	2	1
STEEL	0.5	1.0	1.0	0.5	1.0	1.0	0	2
FIRE	0.5	1.0	0.5	0.5	1.0	1.0	0	3
WATER	2.0	1.0	0.5	2.0	1.0	1.0	2	1
GRASS	0.25	1.0	0.25	0.5	0.5	1.0	0	4
ELECTRIC	2.0	1.0	1.0	1.0	2.0	1.0	2	0
PSYCHIC	1.0	1.0	1.0	2.0	0.5	0.5	1	2
ICE	1.0	1.0	4.0	0.5	2.0	1.0	2	1
DRAGON	1.0	1.0	2.0	1.0	1.0	1.0	1	0
DARK	1.0	1.0	1.0	0.5	2.0	2.0	2	1
FAIRY	0.5	1.0	2.0	1.0	1.0	1.0	1	1

Tips to build a better team:

- The numbers underneath your pokemon's name are the damage multipliers that occur when your pokemon is hit with an attack of the type of that row.
- A multiplier of 1.0 means the attack will deal the normal amount of damage.
- A multiplier of 2.0 means that the attack will deal twice the amount of normal damage. Your pokemon will be weak to attacks of this type
- A multiplier of 0.5 means that the attack will deal half the amount of normal damage. Your pokemon will be resistant to attacks of this type
- A multiplier of 0 means that the attack will deal 0 damage. Your pokemon will be immune to attacks of this type
- Aim to have a Total Weak: Total Resist ratio where the number of Resistances is greater than the number of weaknesses
- Compensate for one pokemon's weakness by including another pokemon that is resistant or immune to that weakness

The more balanced your team is, the better you can handle anything anyone throws at you!

click here to go back to the main page.

128.239.2.105:1738/myapp/?pokemon1=charizard&pokemon2=bidoof&pokemon3=dragonite&pokemon4=emboar&pokemon5=lugia&pokemon6=mewtwo - Google Chrome
Train On.
11:30 AM