Python Promethazine

Justin Chen: HTML/Bootstrap/JS

Alex Olteanu: HTML/Python/Database

Alex Thompson: API/Python/JS

David Wang: PM, Database/JS

Front End Framework: Bootstrap

Topic: Pokemon Showdown

Description (for now):

- A game that's similar to Showdown. Users can build their own team or use a preexisting one and fight the computer in Pokemon battles. We will build a bare version and add features as necessary.
- Limited to Gen 1 Pokemon
- No items/abilities
- No Ditto

APIs:

- Pokemon API

Front End:

1. base.html
    a. Template page
2. landing.html
    a. First page user sees when they enter the site
    b. Options for log-in and register
3. login.html
4. register.html
    a. Only if the user needs to register
5. home.html
    a. User is brought to this page after the user logs in or registers
    b. Options to battle or build teams
6. build.html
    a. Team creator page
    b. If a team is to be created, players must add 6 pokemon at a time
    c. For each pokemon, they must pick 4 moves that the pokemon can learn
    d. Go back to home button when done creating teams
7. setupBattle.html
    a. Choose 2 team(s) that you have already created to battle against each other
    b. Takes you to pvp.html
8. pvp.html (uses JS)
    a. Local 2 player battle, players will alternate turns
    b. Uses saved teams or randomly generates one depending on user selection (per player)
    c. Each turn, the player has an option to select a move unless conditions do not allow it (ex if a pokemon is charging a move, it cannot attack)
    d. First player to have their entire party knocked out loses
    e. Go back to home button when battle completes

Back End:

1. Database
    a. Table: credentials

Username (Text)	Password (Text)
example	example
username	password

    b. Table: teams
        i. 1 table per user, named after username
        ii. Every consecutive 6 pokemon is 1 team

Pokemon (Text)	Move 1 (Text)	Move 2 (Text)	Move 3 (Text)	Move 4 (Text)
bulbasaur	tackle	growl	vine whip	growth

    c. Table: pokemon
        i. 1 table per pokemon, named after pokemon
        ii. Entries are moves that the pokemon is capable of using

Name (Text)
Razor wind
etc

    d. Table: moves
        i. Table that stores every move in this version of the game
        ii. id (integer), name (text), effect (text), damage_type (text), type (text), pp (integer), power (integer), accuracy (integer)

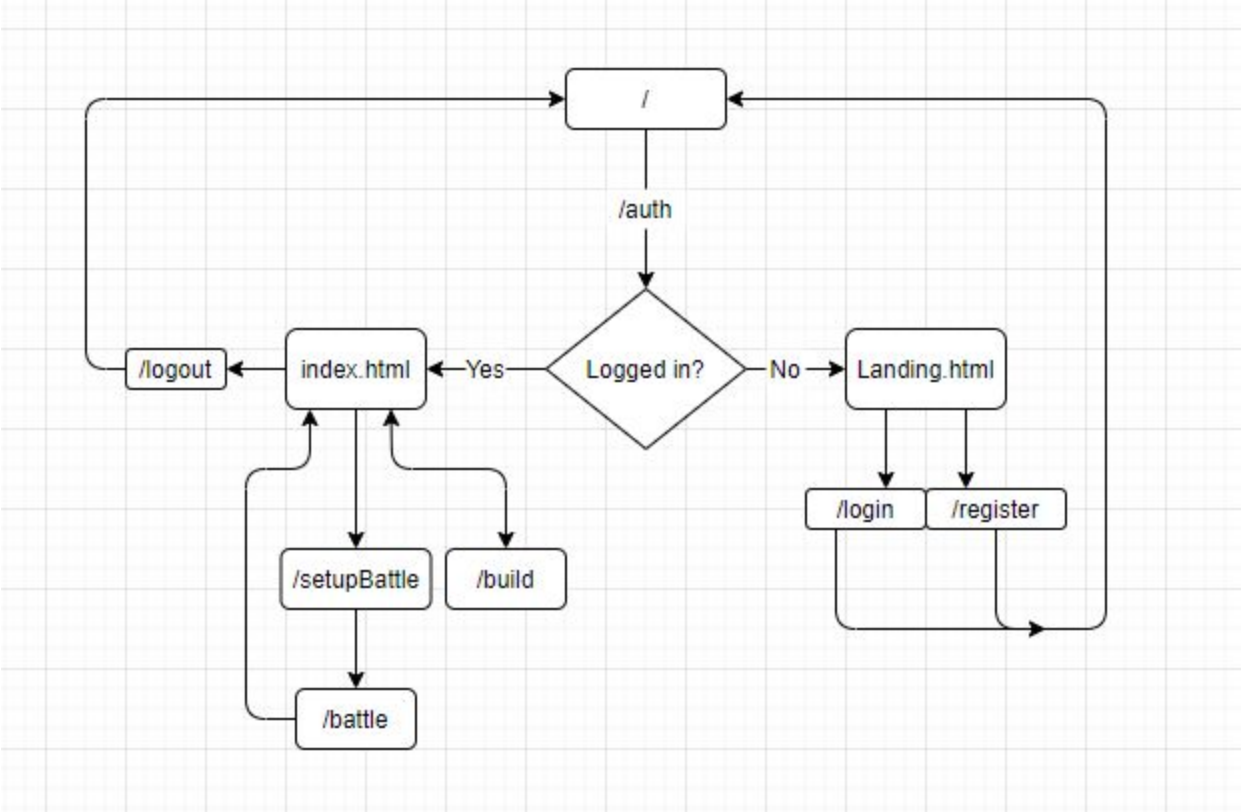
id	name	eff	dcl	type	pp	power	acc
1	pound	Inflicts regular damage...	physical	normal	35	40	100

2. app.py
    a. /
        i. renders home.html if logged in, else landing.html
    b. /login
        i. renders login.html
        ii. redirect to /
    c. /register
        i. renders register.html
        ii. redirect to /
    d. /build
        i. renders build.html
    e. /battle
        i. renders battle.html
    f. /setupBattle
        i. Renders setup.html/set
    g. /logout
        i. redirects to /
        ii. removes user from session

    h. /auth
        i. checks if user is in session, unviewable to user
3. Functions
    a. login()
        i. param: username
        ii. param: password
    b. register()
        i. param: username
        ii. param: password
        iii. All usernames must be unique

c. login()
    i. Renders home.html
d. auth()
    i. param: username
    ii. Checks if user is in session
e. battle()
    i. param: team1
    ii. param: team2
    iii. Renders battle.html
f. build()
    i. Param: username
    ii. Renders team.html
    iii. Add 1 team comprised of 6 Pokemon with 4 moves each to the user's username database
g. logout()
    i. Remove user from session
    ii. Render landing.html
h. attack(player, target, move)
    i. Attacks with specific move stats and calculates the result using the pokeAPI
i. runsqlcommand(command)
    i. Param: command
    ii. Runs the sql command provided
j. cacheMoves()
    i. Caches move data, 1 time use
k. cachePokemon()
    i. Caches pokemon data, 1 time use

Site Map:



Component Map: