Chewy Chupucabras: Ethan Sie, Brian Liu, Tanzeem Hasan
Softdev
P02: Makers Makin' It, Act I
2025-1-3
Time Spent: 4 hours
TARGET SHIP DATE: 2025-01-20

Project Description
This project is a recreation of Pokemon Showdown (some might say better). Users will be able to battle other users with Chupamon (Chupa + Pokemon). Upon logging in, users are automatically entered into an active session list, where any other active user may challenge someone on the list. If the challenge is accepted, then both players will be placed in a battle with six random Pokemon each that follow standard Pokemon rules (users can choose a move on their turn, with the turn order based on Pokemon speed, etc). We will be simplifying the game by removing abilities, items, etc. A ChupaDex will be provided that displays information about all the Pokemon available. Users can also view their match history and a leaderboard.

Frontend Components:

1. **Flask Routes (with Jinja templating) - Updated as new data is requested by python**
    a. **/**
        i. Allows user to register, login, or logout
        ii. Allows user to challenge other users (those that have an active session)
            1. Challenge request is recorded in gameChallenge table
            2. Challenge request can be denied
            3. If accepted, both users' page will be rendered with game.html
        iii. Renders home.html template
    b. **/register**
        i. Allows user to register with password confirmation
        ii. Must have a unique username
        iii. Redirects to / -> renders home.html
    c. **/login**
        i. Allows user to play our game (cannot play if not logged in)
        ii. Cannot log in if the account is already in an active session
        iii. Redirects to / -> renders home.html
    d. **/logout**
        i. Redirects to / -> renders home.html
    e. **/chupadex**
        i. Displays sprites (images) and stats of each Gen 1 Chupamon
            1. Displays a GIF through GIPHY API whenever a pokemon is clicked
            2. If time permits, we will implement a search/filter option
        ii. Renders chupadex.html
    f. **/ladder**
        i. Displays stored leaderboard data based on victories / losses
        ii. Renders ladder.html

- **g. /game**
    - i. Takes the user into our turn-based responsive page
        - 1. Once they have gone through the home.html and successfully challenged…
            - a. Both users select a move
            - b. Damage calculations are done and distributed to your respective pokemon
            - c. Continues until one user is victorious
    - ii. Renders game.html
- **h. /history**
    - i. Displays match history of the user
        - 1. Must be logged in for match history to process
    - ii. Renders history.html

2. **Tailwind CSS - Frontend Framework**
    - a. Tailwind CSS allows the writer to make use of existing utility classes as a shorthand when directly styling elements in HTML.
    - b. Tailwind has built in support for a responsive design, making it easier to create aesthetic buttons and sliders for this project.

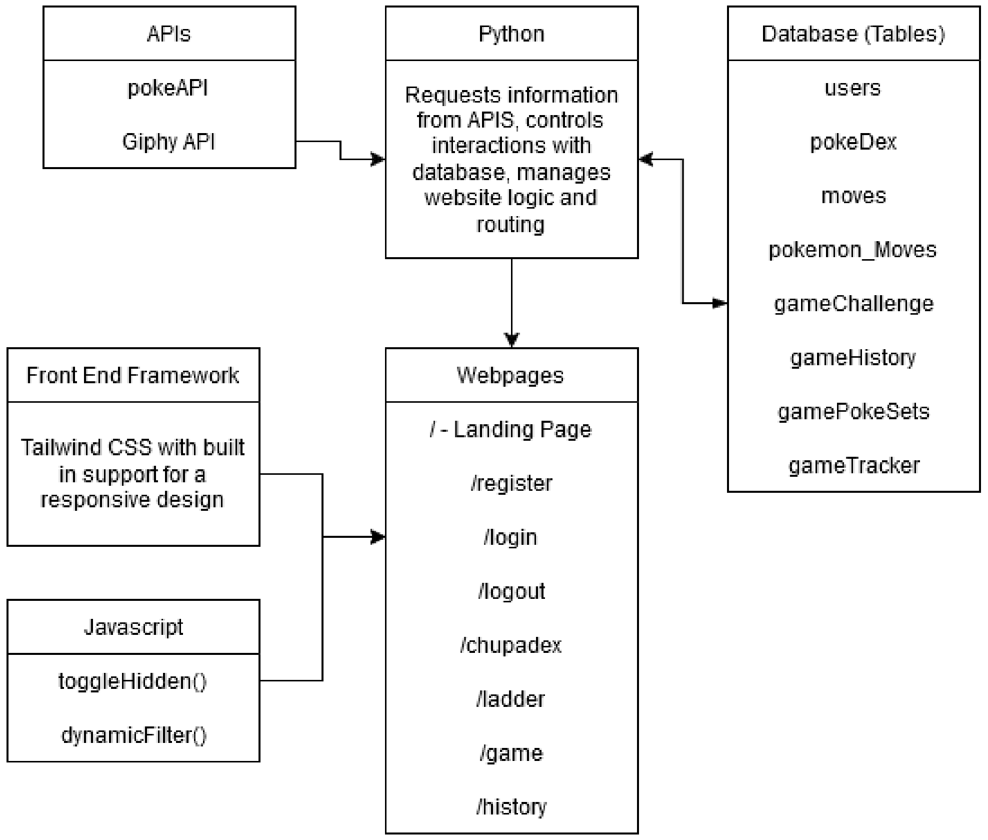
3. **Javascript**
    - a. toggleHidden():
        - i. Toggles show/hide of certain windows (like login, logout, register)
        - ii. Will allow login, logout, and register to happen on the home page instead of rendering a separate template
    - b. dynamicFilter():
        - i. Adjusts the displayed chupadex entries after receiving a filter condition (such as typing)
    - c. More to be added as we go along

Backend Components:

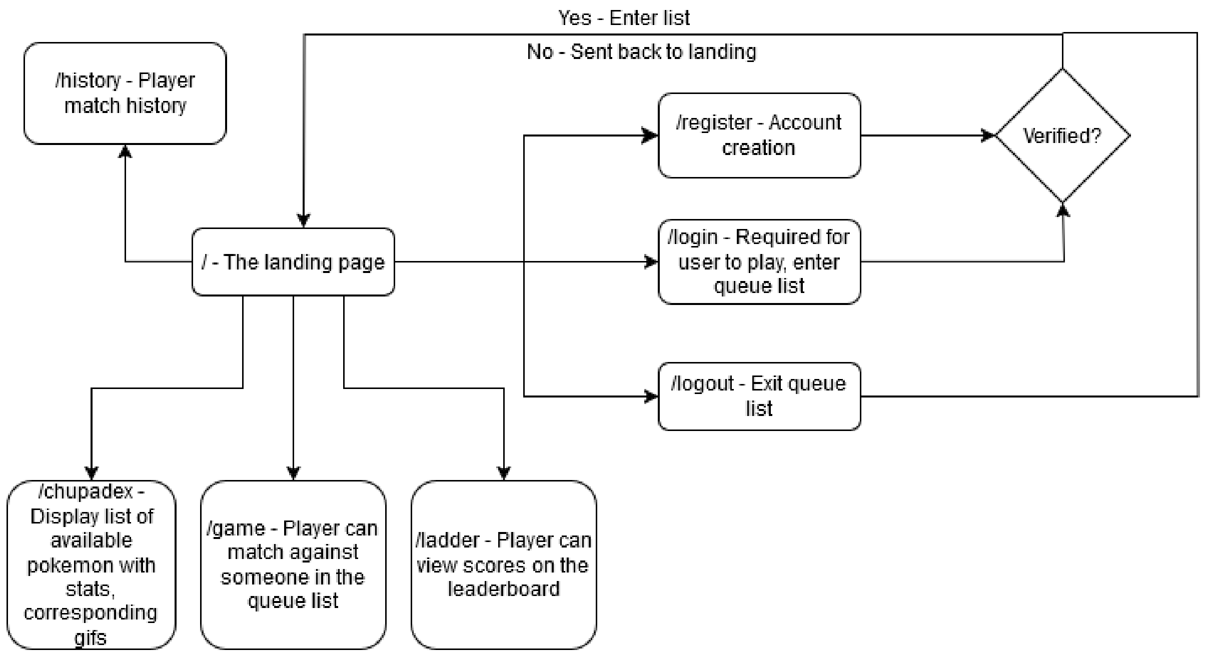
1. Flask/Python
    - a. Allows the user to traverse different web pages
    - b. Python will be used to access specified chupamon data from pre-loaded API data
    - c. We will code the backend for the turn-based system and game components of Chupamon Showdown
    - d. All data will be stored through python into a relevant database
2. SQLite Databases - Stores information from APIs requested by Python
    - a. users: will store user identification, password, name, rank, and last login information
    - b. pokeDex: will store detailed info about each pokemon
    - c. moves: will store information about all Pokemon's moves
    - d. pokemon_moves: stores information about what Pokemons can learn what move
    - e. gameChallenge: will store info about the games being challenged
    - f. gameHistory: will store info about games between players, including winner, loser, and time

g. gamePokeSets: will store the six pokemon each player has as well as their HP values.
h. gameTracker: stores turn info in a game, including turn numbers and moves chosen

Component Map:



Site Map:

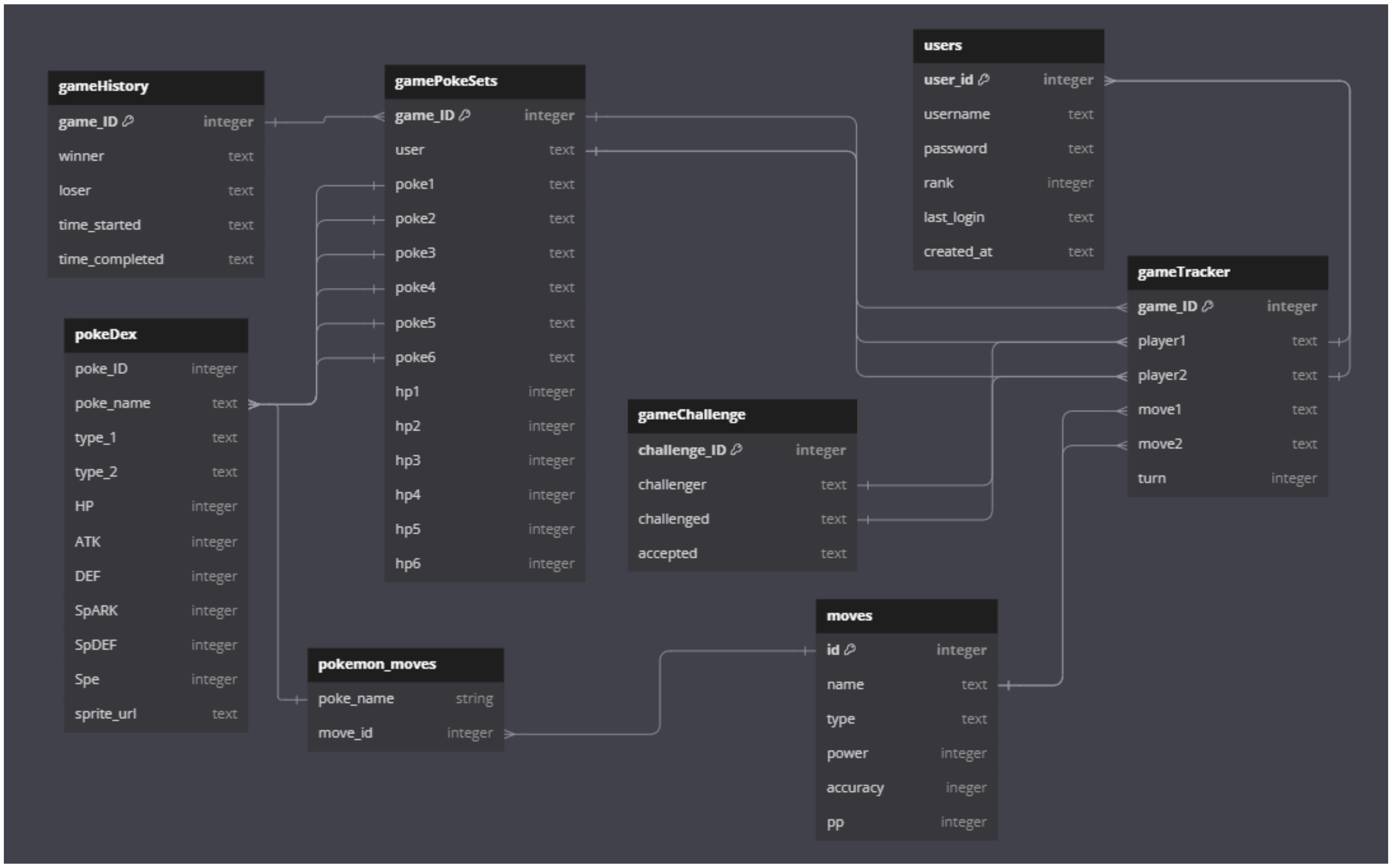

Yes - Enter list
No - Sent back to landing
/history - Player match history
/register - Account creation
Verified?
/ - The landing page
/login - Required for user to play, enter queue list
/logout - Exit queue list
/chupadex - Display list of available pokemon with stats, corresponding gifs
/game - Player can match against someone in the queue list
/ladder - Player can view scores on the leaderboard

Database Organization:

1. users table
    a. user_id (integer): unique identifier per user
    b. username (string): username chosen by user
    c. password (string): hashed password for security
    d. rank (integer): tracks user's rank
    e. last_login (string {date-time}): tracks last user interaction
    f. created_at (string {date-time}): tracks time of creation
2. pokeDex table
    a. poke_ID (integer): identifier for each pokemon
    b. poke_name (string): name of the pokemon
    c. type_1 (string): primary type of the pokemon
    d. type_2 (string): secondary type of the pokemon
    e. HP (integer): base HP of the pokemon
    f. ATK (integer): base attack of the pokemon
    g. DEF (integer): base defense of the pokemon
    h. SpATK (integer): base special attack of the pokemon
    i. SpDEF (integer): base special defense of the pokemon
    j. Spe (integer): base speed of the pokemon
    k. sprite_url (string): url for the pokemon's sprite
3. moves table
    a. id (integer): identifier of each move
    b. name (string): name of the move
    c. type (string): type of the move (ex. fire)
    d. power (integer): base power of the move
    e. accuracy (integer): base accuracy of the move
    f. pp (integer): power points for the move
4. pokemon_moves table
    a. poke_name (string): name of the pokemon
    b. move_id (integer): identifier of a move the pokemon can learn
5. gameChallenge table
    a. challenge_ID (integer): identifier for each challenge
    b. challenger (string): username of the user challenging
    c. challenged (string): username of the challenged user
    d. accepted? (string): whether the challenge was accepted
6. gameHistory table
    a. game_ID (integer): identifier for each game
    b. winner (string): username of the winner of the game
    c. loser (string): username of the loser of the game
    d. time_started (string {date-time}): timestamp of the start of the game
    e. time_completed (string {date-time}): timestamp of the end of the game
7. gamePokeSets table
    a. game_ID (integer): identifier for each game
    b. user (string): username of the player
    c. poke1 (string): name of the user's first pokemon
    d. poke2 (string): name of the user's second pokemon
    e. poke3 (string): name of the user's third pokemon
    f. poke4 (string): name of the user's fourth pokemon
    g. poke5 (string): name of the user's fifth pokemon
    h. poke6 (string): name of the user's sixth pokemon
    i. hp1 (integer): HP of the user's first pokemon
    j. hp2 (integer): HP of the user's second pokemon
    k. hp3 (integer): HP of the user's third pokemon
    l. hp4 (integer): HP of the user's fourth pokemon
    m. hp5 (integer): HP of the user's fifth pokemon
    n. hp6 (integer): HP of the user's sixth pokemon
8. gameTracker table

a. game_ID (integer): identifier for each game
b. player1 (string): username of player1
c. player2 (string): username of player2
d. move1 (string): move chosen by player1
e. turn (integer): current turn number in the game

Database Diagram (using dbdiagram.io)



APIs to use:

- PokeAPI:
  - API: https://pokeapi.co/
  - Github Card: https://github.com/stuy-softdev/notes-and-code/blob/main/api_kb/411_on_pokeapi.md
  - Provides access to Pokemon data
  - No key + no quotas
- Giphy API
  - API: https://developers.giphy.com/docs/
  - Github Card: https://github.com/stuy-softdev/notes-and-code/blob/main/api_kb/411_on_GiphyAPI.md
  - Provides GIFs for Chupamons upon selection in the ChupaDex
  - Requires key

Tasks:

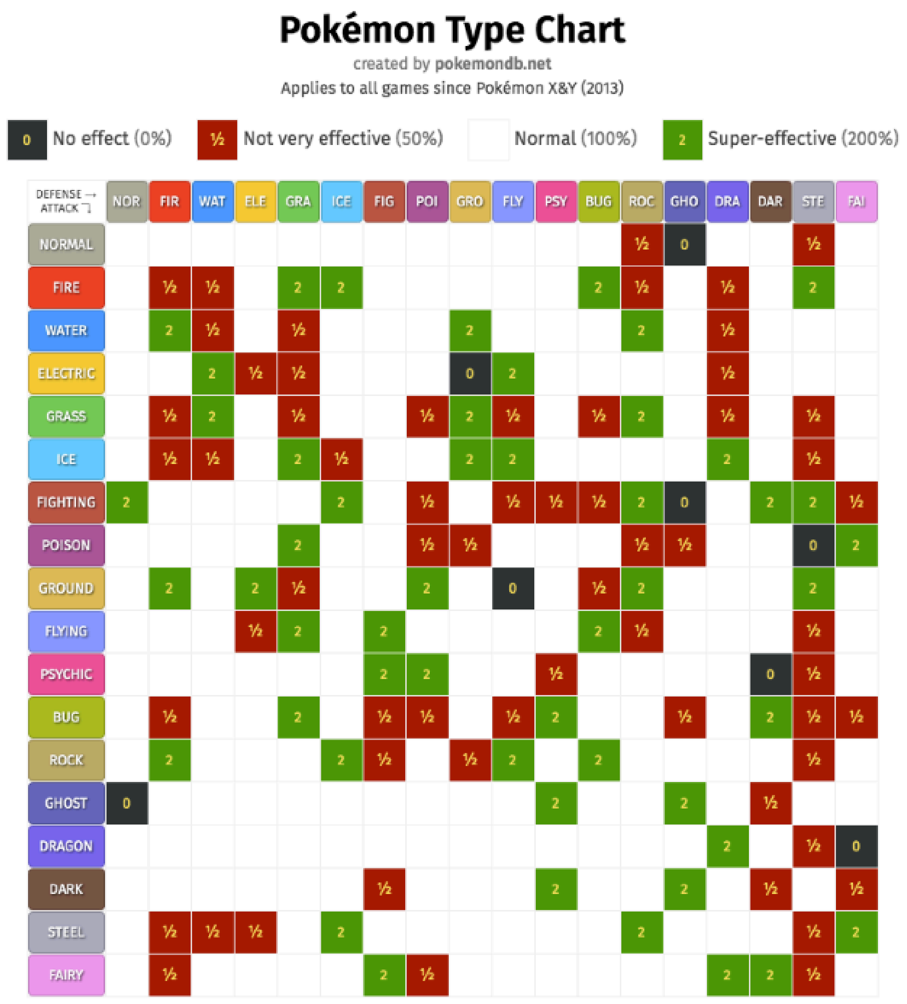
Ethan Sie:

- Database lead
- Basic site setup
    - Creating base tailwind for overall site design
    - SQLite db setup
    - Ensures all program file templates are accessible and ready for use
- Create the matching system through a dictionary of tracked active sessions
- Ensure turn-based system is working through database-tracked turns/games
- Will help to catch up with any other work

Tanzeem Hasan:
- Javascript lead for more intricate animations and website capabilities
    - Dynamic response design in the game page
    - Working with Ethan to create the mechanics behind the actual game component
- Ensure cleanliness of database code (as we are spending a lot of time populating the db)
- Determine settings that can be toggled in account settings
    - Possibly creating light/dark modes or more user customization to the site design

Brian Liu:
- Tailwind (css) lead for more intricate designs
    - The game page
    - Ladder (ranking) page
    - Handling dialogue output throughout the matches
- Handling damage calculations
- Creating a system for point calculations (elo) based on wins/loss, pokemon alive, etc
- Designing and adding responsiveness to match history page

Resources:



Pokémon Type Chart

created by pokemondb.net

Applies to all games since Pokémon X&Y (2013)

0 No effect (0%) ½ Not very effective (50%) Normal (100%) 2 Super-effective (200%)

DEFENSE → ATTACK ↴	NOR	FIR	WAT	ELE	GRA	ICE	FIG	POI	GRO	FLY	PSY	BUG	ROC	GHO	DRA	DAR	STE	FAI
NORMAL													½	0			½	
FIRE		½	½		2	2						2	½		½		2	
WATER		2	½		½				2				2		½			
ELECTRIC			2	½	½				0	2					½			
GRASS		½	2		½			½	2	½		½	2		½		½	
ICE		½	½		2	½			2	2					2		½	
FIGHTING	2					2		½		½	½	½	2	0		2	2	½
POISON					2			½	½				½	½			0	2
GROUND		2		2	½			2		0		½	2				2	
FLYING				½	2		2					2	½				½	
PSYCHIC							2	2			½					0	½	
BUG		½			2		½	½		½	2			½		2	½	½
ROCK		2				2	½		½	2		2					½	
GHOST	0										2			2		½		
DRAGON															2		½	0
DARK							½				2			2		½		½
STEEL		½	½	½		2							2				½	2
FAIRY		½					2	½							2	2	½	

Generation I

$$Damage = \left(\frac{\left(\frac{2 \times Level \times Critical}{5} + 2 \right) \times Power \times A/D}{50} + 2 \right) \times STAB \times Type1 \times Type2 \times random$$

where:

- *Level* is the level of the attacking Pokémon.
- *Critical* is 2 for a critical hit, and 1 otherwise.
- *A* is the effective Attack stat of the attacking Pokémon if the used move is a physical move, or the effective Special stat of the attacking Pokémon if the used move is a special move (for a critical hit, all modifiers are ignored, and the unmodified Attack or Special is used instead). If either this or *D* are greater than 255, both are divided by 4 and rounded down.
- *D* is the effective Defense stat of the target if the used move is a physical move, or the effective Special stat of the target if the used move is an other special move (for a critical hit, all modifiers are ignored, and the unmodified Defense or Special is used instead). If the move is physical and the target has Reflect up, or if the move is special and the target has Light Screen up, this value is doubled (unless it is a critical hit). If the move is Explosion or Selfdestruct, this value is halved (rounded down, with a minimum of 1). If either this or *A* are greater than 255, both are divided by 4 and rounded down. Unlike future Generations, if this is 0, the division is not made equal to 0; rather, the game will try to divide by 0 and softlock, hanging indefinitely until it is turned off.
- *Power* is the power of the used move.
- *STAB* is the same-type attack bonus. This is equal to 1.5 if the move's type matches any of the user's types, and 1 if otherwise. Internally, it is recognized as an addition of the damage calculated thus far divided by 2, rounded down, then added to the damage calculated thus far.
- *Type1* is the type effectiveness of the used move against the target's type that comes first in the type matchup table, or only type if it only has one type. This can be 0.5 (not very effective), 1 (normally effective), 2 (super effective).
- *Type2* is the type effectiveness of the used move against the target's type that comes second in the type matchup table. This can be 0.5 (not very effective), 1 (normally effective), 2 (super effective). If the target only has one type, *Type2* is 1. If this would result in 0 damage, the calculation ends here and the move is stated to have missed, even if it would've hit.
- *random* is realized as a multiplication by a random uniformly distributed integer between 217 and 255 (inclusive), followed by an integer division by 255. If the calculated damage thus far is 1, *random* is always 1.