Multiplayer Version 1.0 Design

- Interaction with Team Selection
    - User has a team selected, with their data stored in the database
        - See explanation at the end as to why the battle team shouldn't be ultimately just stored in $_SESSION
    - User is then directed to the *find_match.html* page.
- Internal Processing (independent)
    - The page simply displays "Finding Match"
    - On page load, it fires off an api call *find_match* periodically, once every X seconds. It does this until the user clicks "stop", back, or a "match_found" reply is received.
    - On the back end, the *find_match* php function is called. It adds if the user's secret to a "active_users" list or verifies that the secret is already there.
    - The "active_users" list can be stored in either shared memory (https://php.net/manual/en/book.memcached.php) or in the database.
    - *find_match* then scans the list for active users. It will know that a user is active through one of two ways (currently option 2):
        - Option 1: A persistently running php script, *maintain_active_users.php*, would delete users from the list that have not call *find_match* within a certain time period (once every X seconds).
        - Option 2: The list would store users and a timestamp indicating the last time they called *find_match*. If the timestamp is within X seconds of the current time, then the user is matched as an opponent.
    - If an active user is found, then *find_match* first checks shared data (database) to see if they have already been matched (this helps stop the script from making multiple matches and trying to start multiple battles simultaneously).
    - If not already matched, then *find_match* matches the user with the found user. If either the user is found to be matched, or a match is initiated with a found user, the api_call responds to the front end with "match_found".
    - Both users are then asked to confirm the match in a pop-up window or intermediate page

        - This confirmation means that bad matches fail here, instead of in the battle screen, which should help make debugging easier
- Interaction with Battle Screen
    - Whichever instance of the *find_match* function initiated the match now takes responsibility for starting up the battle screen, and is designated as Player 1, while the other instance/player is designated as Player 2
    - The Player 1 instance of *find_match* calls php functions (from Andres's battle mechanics code) to construct the GameState using their and their opponent's battle teams
        - The battle teams thus have to be stored in cross-session storage so that both players can access their opponent's battle team
        - The GameState must also be stored in cross-session storage, so that it can be updated during battle when an opponent makes a move.
    - The battle screen page (*battle.html*?) is then launched. Each player's front end loads the current GameState periodically, and modifies the shared GameState with the battle mechanics api code.

<u>Schema Adition (V 1.0):</u>

- Teams (team_id, username, creation_timestamp, animal_1, animal_2, animal_3, animal_4, animal_5, animal_6)
    - username and all animals are foreign key constraint
    - team_id is the auto_incremented primary key
    - the timestamp might be useful for recency (e.g. the default team is the last used one)
- Matches (match_id, player1_username, player2_ username, player1_team, player2_team, creation_timestamp, winner_username)
    - The usernames are foreign key constraints.
    - The team_ids are foreign key constraints.
    - match_id is the auto_incremented _primary key
    - The winner_username is NULL if the battle was never finished
    - An additional "status" row might be useful (e.g. "match found", "battle initiated", "battle concluded") for debugging, but isn't necessary for now
- GameStates (match_id, super long string game_state)

- Waiting(player_username, creation_timestamp)
    - The player username is foreign key constraint
    - The player_username and timestamp combination is the primary key. Didn't add an id here, don't think we'd ever need to refer to a row in this table.