

CS 2300 Database Project

Phase III

Jack Kufa

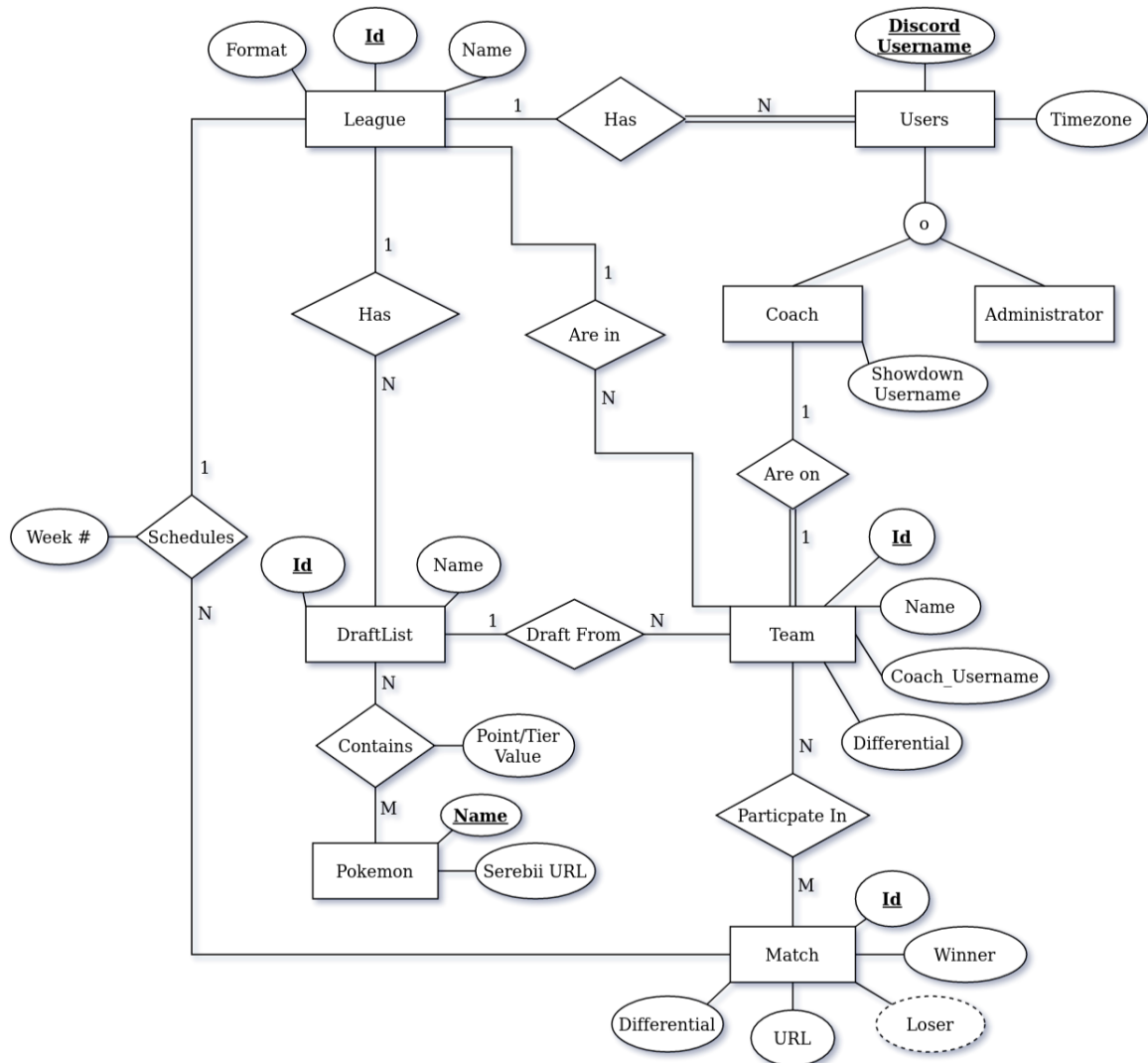
December 12, 2020

Problem Statement

I build a Discord bot and localhost web application for managing and automating a Pokemon Draft League. A Pokemon Draft League is a custom game mode for Pokemon battling, where coaches form teams and draft Pokemon to compete head to head. This application is able to automate a lot of the internal work involved in running such a league and it streamlines specific aspects of league upkeep such as updating rankings.

The core interface is the Discord Bot. This is because Discord is a vital platform for communication between players, and being able to query data in that same space is very convenient. With that said, the web application, built for administrative use, is perfect to getting a league up and running.

ER Model



The six primary entity sets are as follows:

1. League: The Main entity that specifies the battle format and draft list being used. Essentially every other entity is related to the league in some capacity.
2. Users: Users who are participating in the league. The subclasses are as follows:
 - (a) Coach: Think of it like the coach of a sports team. These are the players that are actually participating in the league.
 - (b) Administrator: A person who has additional privileges for managing the league.

* It should be noted that a user can be either a coach or an administrator, or both, but it cannot be neither.

3. Teams: Another big entity which is essentially an equivalent of a sports team. It has a team name, coach username, and differential. The differential is an integer value that equates to the total number of Pokemon beaten in any match.
4. Pokemon: This entity is used to represent the creatures used in battle; think of them as the players for a sports team. Each Pokemon has a name, and for convenience of the user, a url to the Serebii page, which has important information such as Pokemon Type, move set, and stats. A draft list contains Pokemon, associated each with a point value.
5. DraftList: This is the reference sheet used for drafting Pokemon. Each Pokemon in a draft list has an associated value that indicates its "cost" in the draft league, including a list of banned Pokemon.
6. Match: This entity is for storing information related to Pokemon battles. Matches are played on Pokemon Showdown. This entity stores the match's url, differential (The difference in knocked out Pokemon on either side), and winner, meaning that subsequently a loser can also be derived.

There are some aspects of this entity relationship diagram that should be noted. First, any tables that potentially would have contained multivalued attributes, such as a team's Pokemon, were optimized to 1NF specifications so that no multivalued attributes were actually needed. Second, there is a redundant relationship that exists, the relationship between the league and teams. This relationship was added due to time constraints, for there were times where querying a team based on its league proved to be infinitely more convenient than querying users first. Ideally, however, this is a relationship that is unnecessary.

Logical Database Design

Summary of Data Types

Table	Attribute	Type	Constraint
League	Id	INTEGER	Primary Key
League	Name	VARCHAR(256)	Unique
League	Format	VARCHAR(20)	
League	Dlist_id	INTEGER	Foreign Key
User	Username	VARCHAR	Primary Key
User	Timezone	CHAR(6)	
User	League_id	INTEGER	Foreign Key
Coach	Discord_username	VARCHAR	Foreign Key
Coach	Showdown_username	VARCHAR	Unique
Administrator	Discord_Username	VARCHAR	Foreign Key
Team	Id	INTEGER	Primary Key
Team	League_id	INTEGER	
Team	Name	VARCHAR(80)	Unique
Team	Differential	INTEGER	
Team	Coach_username	VARCHAR	Foreign Key
DraftList	Id	INTEGER	Primary Key
DraftList	Name	VARCHAR(50)	Unique
Pokemon	Name	VARCHAR(25)	Primary Key
Pokemon	Url	VARCHAR(60)	
Match	Id	INTEGER	Primary Key
Match	Week_no	INTEGER	
Match	Differential	INTEGER	
Match	Url	VARCHAR(60)	
Match	Winner	VARCHAR(80)	
Match_League	league_id	INTEGER	Primary Key, Foreign Key
Match_League	Match_id	INTEGER	Primary Key, Foreign Key
DraftList_Pokemon	Pkmn_name	VARCHAR(25)	Primary Key, Foreign Key
DraftList_Pokemon	Dlist_id	INTEGER	Primary Key, Foreign Key
DraftList_Pokemon	Value	VARCHAR(10)	
Pokemon_Team	Pkmn_name	VARCHAR(25)	Primary Key, Foreign Key
Pokemon_Team	Team_id	INTEGER	Primary Key, Foreign Key
Team_Match	Team_id	INTEGER	Primary Key, Foreign Key
Team_Match	Match_id	INTEGER	Primary Key, Foreign Key

Application Program Design (Functionality)

The following is pseudocode for some of the functions that have been implemented. Please note that while the query syntax is made to mimic SQL, it is not 100% the same syntactically, and should not be treated as such.

- BASIC FUNCTIONS:

1. Draft Pokemon (Insert):

```
!draft <Pokemon Name>
function DRAFT_POKEMON(discord_username, user_message)
  if POKEMON contains user_message then
    pokemon = SELECT Name FROM POKEMON
               WHERE Name = user_message
    coach = SELECT Name FROM COACH
             WHERE Name = discord_username
    INSERT INTO POKEMON_TEAM Pokemon_Name, Team_Name
    VALUES pokemon, coach
    print "You selected: " + pokemon
  else print "ERROR. That is not a valid Pokemon!"
  end if
end function
```

2. Submit Replay (Insert):

```
!submit <Replay URL>
function SUBMIT_REPLAY(user_message)
  if user_message starts with https://replay... then
    Parse website data for winner, loser, differential, and id
    INSERT INTO MATCH ID, Differential
    VALUES id, differential
    INSERT INTO MATCH_PLAYERS Winner, Loser
    VALUES winner, loser
  end if
end function
```

3. Redraft Pokemon (Modify):

```

!redraft <Pokemon Name 1> <Pokemon Name 2>
function REDRAFT_POKEMON(discord_username, user_message [ ])
  Split user_message into find and replace
  if POKEMON contains replace then
    find_pokemon = SELECT Name FROM POKEMON
                      WHERE Name = find
    replace_pokemon = SELECT Name FROM POKEMON
                      WHERE Name = replace
    UPDATE POKEMON_COACH
    SET Pokemon_Name = replace
    WHERE Pokemon_Name = find
    print find + " Has been replaced with " + replace
  else print "ERROR. That is not a valid Pokemon!"
  end if
end function

```

4. Delete User (Remove):

```

!delete <username>
function DELETE_USER(user_message)
  Split user_message into find and replace
  if POKEMON contains replace then
    find_pokemon = SELECT Name FROM POKEMON
                      WHERE Name = find
    replace_pokemon = SELECT Name FROM POKEMON
                      WHERE Name = replace
    UPDATE POKEMON_COACH
    SET Pokemon_Name = replace
    WHERE Pokemon_Name = find
    print find + " Has been replaced with " + replace
  else print "ERROR. That is not a valid Pokemon!"
  end if
end function

```

1. Query all of a user's info (join):

```

!userinfo <User>
function USER_INFO(user_message)
  if USER contains user_message then
    R1 = INNER JOIN USER.Discord_Username = COACH.Discord_Username
    R2 = INNER JOIN R1.Discord_Username = COACH_TEAM.Coach_Username
    info = SELECT Discord_Username, Team_Name, Showdown_Name FROM R2
          WHERE Discord_Name = user_message
    print info
  end if
end function

```

2. Calc Average Differential (Join,Average):

```
internal function
function DIFFERENTIAL(team_name)
    positive = SELECT SUM Differential FROM MATCH
                WHERE MATCH.Winner = team_name
    negative = SELECT SUM Differential FROM MATCH
                WHERE MATCH.Loser = team_name
    SELECT AVG Differential FROM MATCH
    return positive - negative
end function
```

3. Query rankings (Order By):

```
!rankings
function RANKINGS
    rankings = SELECT team FROM TEAM
                ORDERBY DIFFERENTIAL(team) + WINS(teams)
                    > DIFFERENTIAL(previous_team) + WINS(previous_teams)

    print rankings
end function
```

4. Query matches played (Join):

```
!matchesplayed
function MATCHES_PLAYED
    matches = SELECT ID FROM MATCH
                WHERE ID! = NULL
    print matches as URL
end function
```

5. Query specific Pokemon:

```
!pokemon <Pokemon Name>
function QUERY_POKEMON(user_message)
    if POKEMON contains user_message then
        pokemon = SELECT Name FROM POKEMON
                    WHERE Name = user_message
        print pokemon
    else print "ERROR. That is not a valid Pokemon!"
    end if
end function
```

Installation Instructions

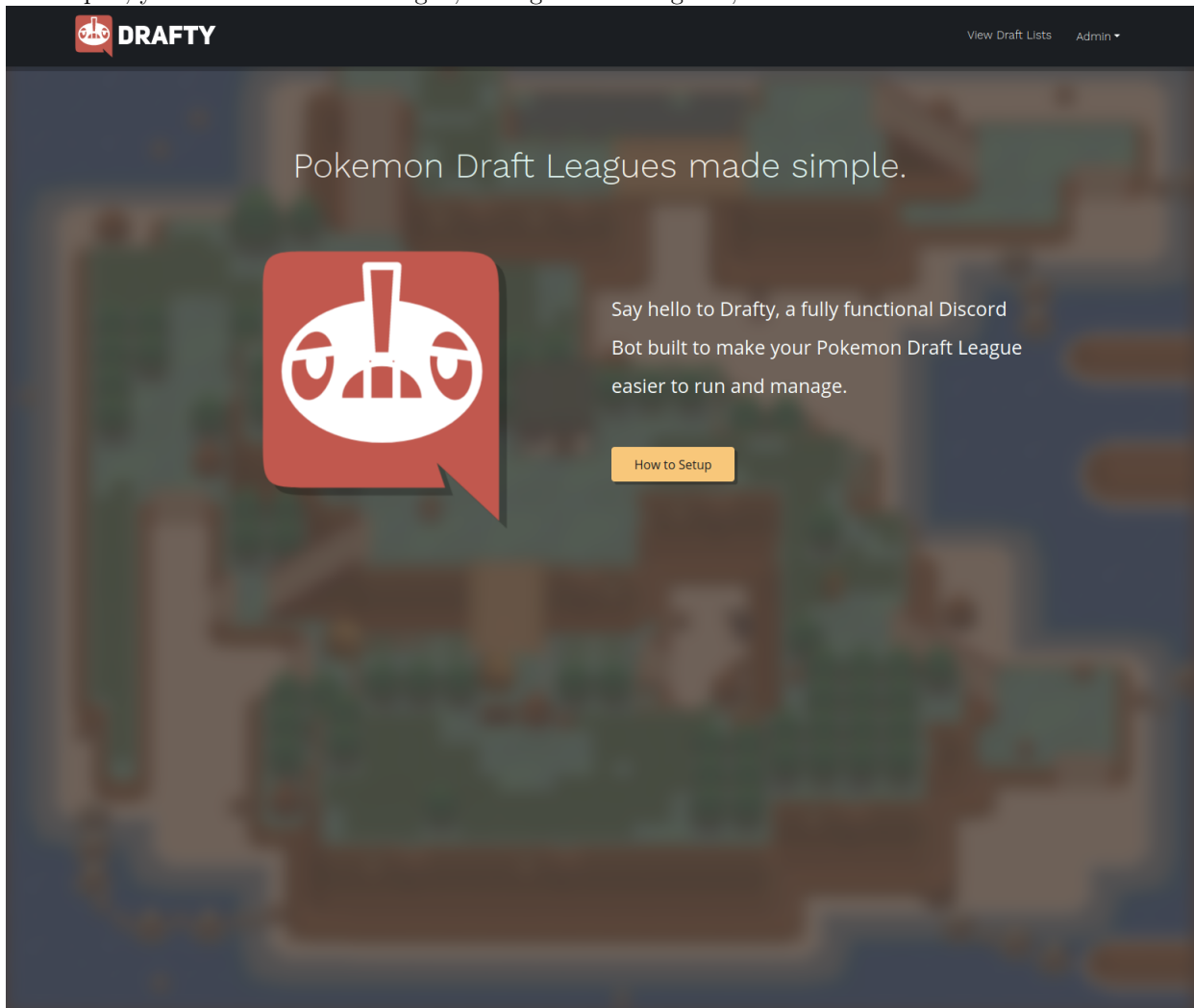
User Manual

Below is documentation related to using the web application and discord bot to interact with the database.

Using the Web Application

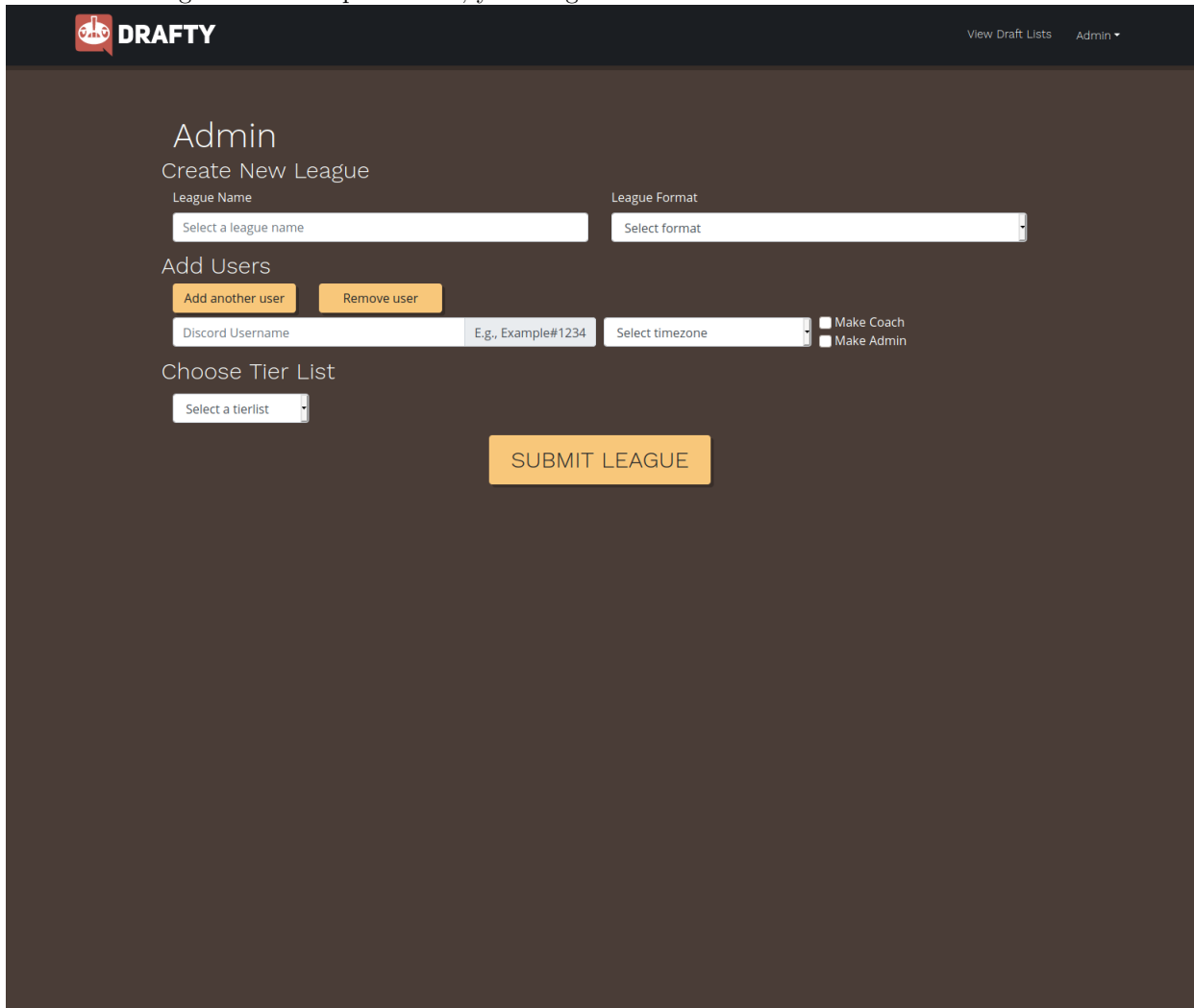
To start the web application, run `deploy_app.sh` to begin the web application, and go to `http://127.0.0.1:5000/` in your web browser. For more details, follow the instructions in Installation Instructions.

Once open, you can create a new league, manage an existing one, or view a draft list.



Create League

To create a league, navigate to the Create League section under Admin on the navbar. Once there, the interface should be relatively straightforward. First, give your league a name, and decide on a format. Currently the only supported format is Gen 8 OU. Next, Add users to the league. Please note you cannot add users later, as the match schedule is determined upon creation of the league. Each user can either be a coach or an admin, or both, but you CANNOT add a user that is neither. An admin gets extra permissions in discord, such as the ability to turn drafting and redrafting on or off. Once all your users have been added, the next step is to select a tier list. Currently, only premade lists are available to use. But later on, the option to import your own will be supported. Last, hit the Submit League Button. Upon reload, your league will have been created!

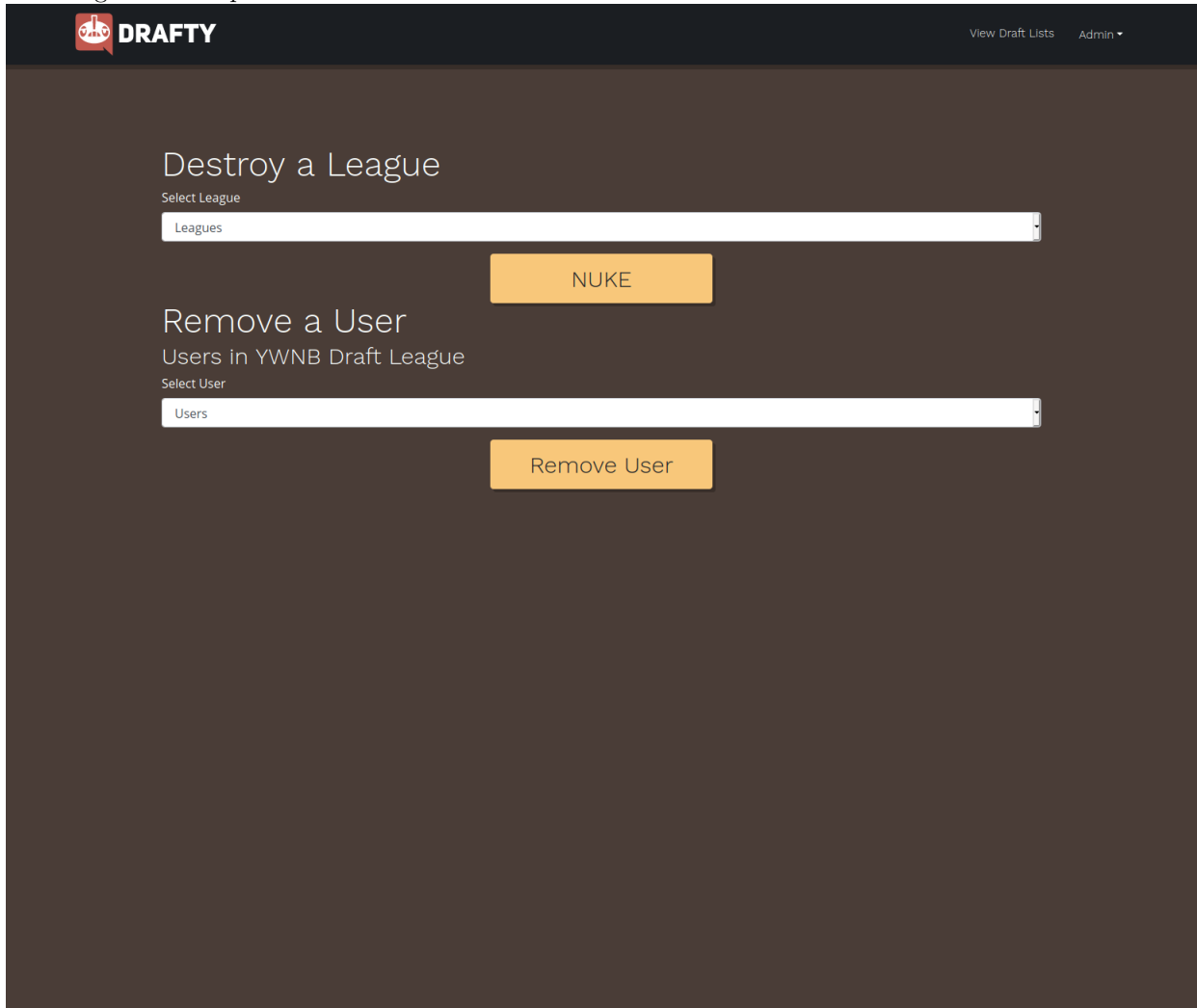


The screenshot shows the 'Admin' section of the Drafty application. At the top, there's a dark header with the 'DRAFTY' logo and a 'View Draft Lists' link. Below the header, the 'Admin' title is displayed. The main section is titled 'Create New League'. It contains several form elements: a 'League Name' field with a placeholder 'Select a league name', a 'League Format' dropdown menu with a placeholder 'Select format', an 'Add Users' section with 'Add another user' and 'Remove user' buttons, a 'Discord Username' field with a placeholder 'E.g., Example#1234', a 'Select timezone' dropdown menu, and two checkboxes for 'Make Coach' and 'Make Admin'. Below these is a 'Choose Tier List' section with a 'Select a tierlist' dropdown menu. At the bottom, there is a large orange 'SUBMIT LEAGUE' button.

Manage League

To manage a league, navigate to the Manage League section under Admin on the navbar. Once there, you will have a couple options. You can either delete an entire league, or delete a user from a

league. Deleting a league deletes all users, all teams, and all matches. Deleting a user delete's that user's team and removes their matches. There will be Remove User section for each league, since each league has unique users.



The screenshot shows the DRAFTY web application interface. At the top, there is a dark blue header with the DRAFTY logo on the left and links for 'View Draft Lists' and 'Admin' on the right. The main content area has a dark brown background. It features two sections: 'Destroy a League' and 'Remove a User'. The 'Destroy a League' section includes a 'Select League' dropdown menu with 'Leagues' selected and a yellow 'NUKE' button. The 'Remove a User' section includes a 'Users in YWNB Draft League' title, a 'Select User' dropdown menu with 'Users' selected, and a yellow 'Remove User' button.

View Draft List


To view a draft list, navigate to the View Draft Lists section. Once there, simply select a league and hit submit to view the draft list being used for that league.

View Draft Lists

Select League

Leagues

View Draft List


[View Draft Lists](#)
[Admin](#)

VIEW DRAFT LIST

Draft List

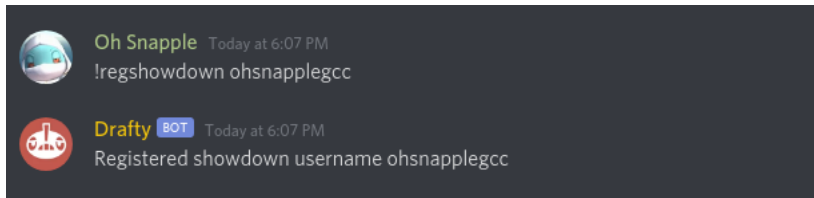
Banned	1	2	3	4	5
Dracovish	Aegislash	Alakazam	Barraskewda	Abomasnow	Appletun
Eternatus	Azumarill	Blissey	Bisharp	Accelgor	Aromatisse
Kyurem-Black	Clefable	Chansey	Blastoise	Alcremie	Basculin
Kyurem-White	Corviknight	Conkeldurr	Celebi	Araquanid	Beartic
Lunala	Dragapult	Ferrothorn	Cloyster	Arcanine	Beheeyem
Magearna	Excadrill	Gengar	Corsola-Galar	Arctovish	Bellossom
Melmetal	Hawlucha	Grimmsnarl	Crawdaunt	Arctozolt	Bouffalant
Mew	Hydreigon	Gyarados	Diggersby	Avalugg	Butterfree
Mewtwo	Jirachi	Hatterene	Dracozolt	Barbaracle	Cherrim
Necrozma-Dawn-Wings	Kommo-o	Heracross	Duraludon	Bewear	Cincino
Necrozma-Dusk-Mane	Kyurem	Hippowdon	Exploud	Boltund	Clawitzer
Reshiram	Necrozma	Indeedge-Male	Haxorus	Braviary	Claydol
Solgaleo	Rillaboom	Keldeo	Incineroar	Bronzong	Clefairy
Urshifu-Rapid-Strike	Terrakion	Kingdra	Indeedge-Female	Centiskorch	Coalossal
Urshifu-Single-Strike	Tyranitar	Magnezone	Inteleon	Chandelure	Corsola
Zacian	Volcarona	Pelipper	Klefki	Charizard	Cramorant
Zamazenta	Zeraora	Scizor	Krookodile	Cobalion	Crustle
Zekrom		Scolipede	Lycanroc-Dusk	Cofagrigus	Dedenne
Dugtrio		Silvally	Lycanroc-Midday	Comfey	Drampa
Gothitelle		Slowbro	Mamoswine	Copperajah	Drifblim
Gothorita		Starmie	Darmanitan	Cursola	Dubwool
Cinderace		Togekiss	Marowak-Alola	Decidueye	Dugtrio-Alola
Darmanitan-Galar		Torkoal	Milotic	Dhelmise	Dunsparce
Nidoran M		Toxapex	Mimikyu	Ditto	Duosion
Nidoran F		Toxtricity	Ninetales	Doublade	Dusclops
		Mandibuzz	Obstagoon	Dragalge	Dusknoir
		Skarmory	Politoed	Drapion	Eldegoss
		Amoonguss	Polteageist	Drednaw	Emolga
		Ninetales-Alola	Porygon-Z	Druidigon	Exeggutor
			Porygon2	Durant	Falinks
			Primarina	Eiscue	Ferroseed
			Raichu-Alola	Escavalier	Flareon

Using The Discord Bot

To start the Discord bot, run `deploy_bot.sh` to begin the web application, and type in the name of the league you would like to run the league for. For more details, follow the instructions in Installation Instructions. The name of the league included in the dummy data is "YWNB Draft League"

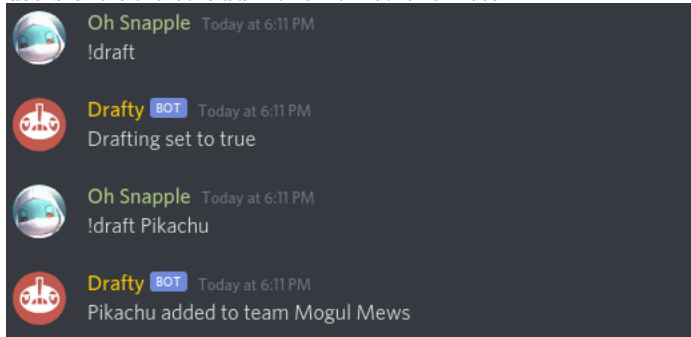
Update showdown username

The first step for all participants in the draft league is to register their showdown username. To do this, users should perform the following command: `!regshowdown ohsnaplegcc`. Upon completion, if no one is already using that showdown username, Drafty will inform you that the update was a success.



Drafting

To enable Pokemon drafting for teams, an admin must first enable it using **!draft**. Once enabled, users are able to add Pokemon to their team.



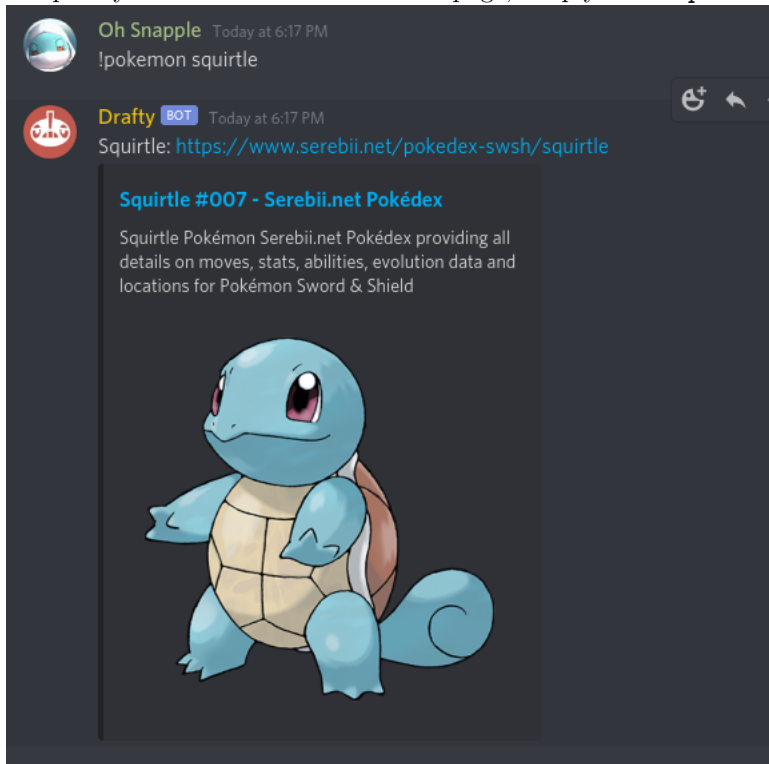
Redrafting

To enable rerafting for teams, an admin must first enable it using **!redraft**. Once enabled, users are able to swap out Pokemon in their team.



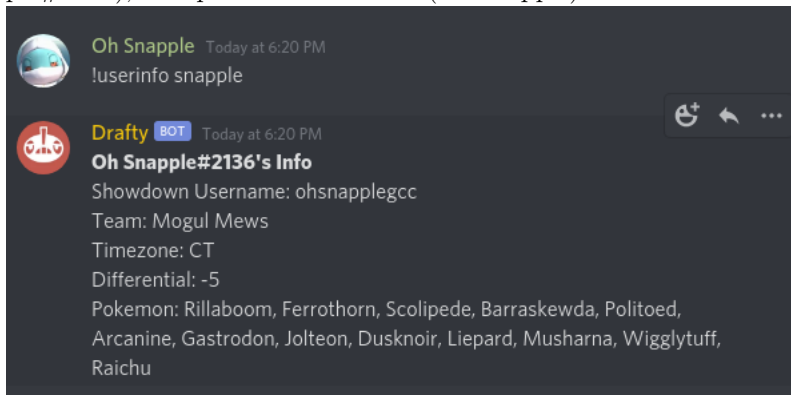
View Pokemon

To quickly access a Pokemon's serebii page, simply enter `!pokemon <pokemon name>`.



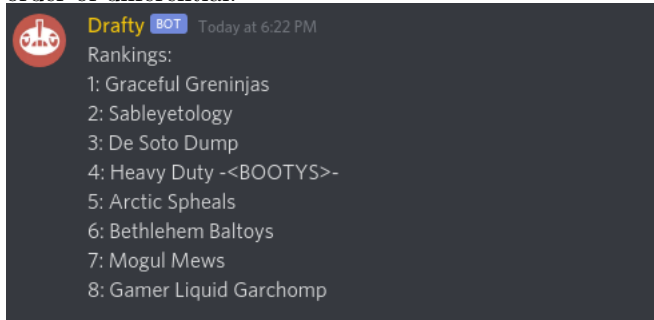
View User Info

To view a user's info, simply enter `!userinfo <user's name>` to view all stats related to that coach and their team. Alternatively, `!userinfo` can be used with no parameters to view the userinfo for yourself. When searching for a user, you can either type their full discord username (ex: Oh Snapple#2136), or a piece of their name (ex: snapple).



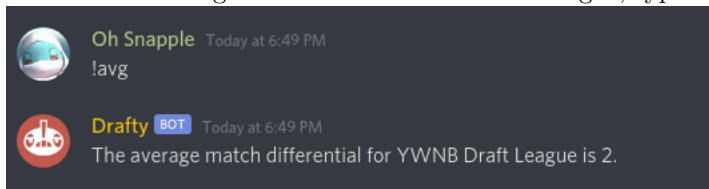
Rankings

To view the rankings in the league, type the command `!rankings`. This will display the teams in order of differential.



Calculate Average Differential

To view the average match differential for the league, type `!average` or `!avg`.



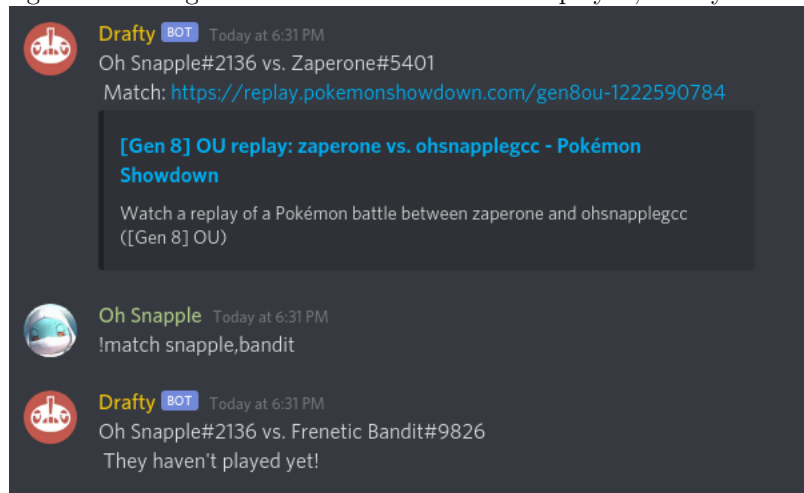
View User's Matches

To view all the matches a user plays in the league, type `!matches <user's name>` or `!matches`. This will display every match the user has to play in weekly order.



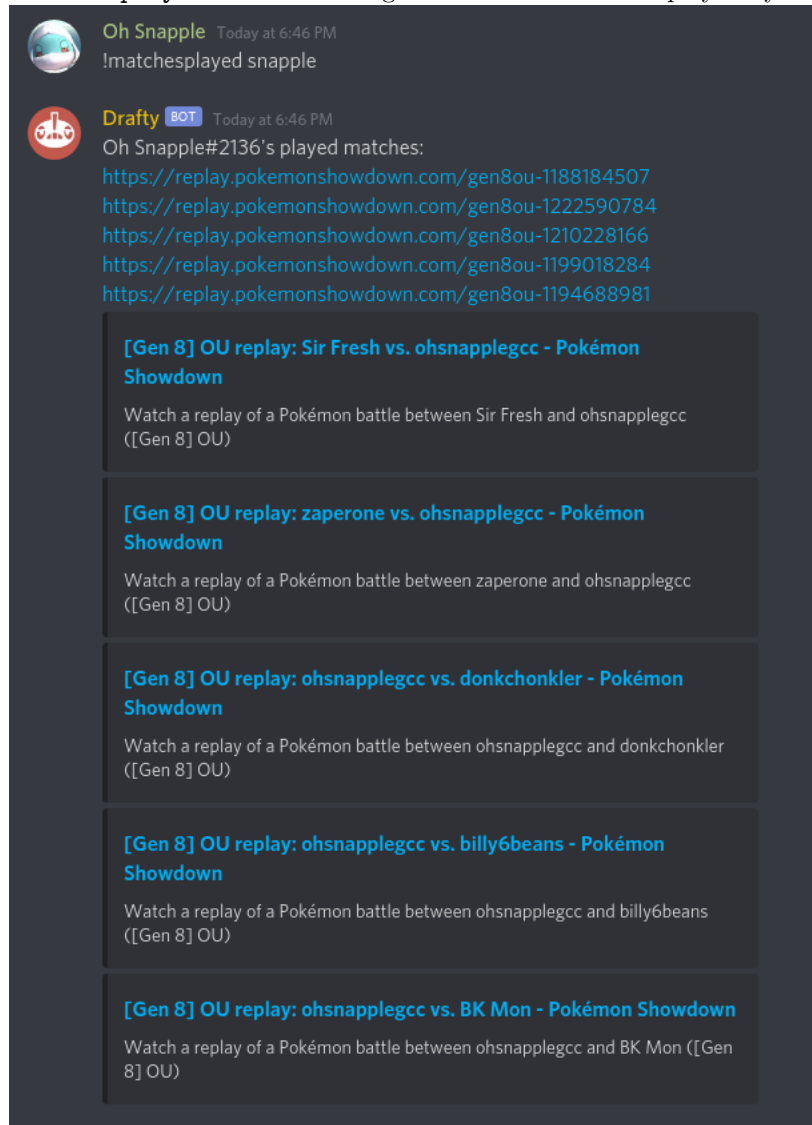
View a Match Played

To find the url for a specific match, type `!match <user1>,<user2>`. **Make sure the usernames are seprated by a comma only.** This is because usernames can include spaces, so it is not a significant distiguisher. If a match hasn't been played, Drafty will let you know.



View User's Matches Played

To get a list of all of a user's played matches, type `!matchesplayed <username>`. Alternatively, `!matchesplayed` can be used to get urls for the matches played by the current discord user.



Delete User

Admins can delete users from the league through Drafty as well, simply type `!delete <discord_username>`. The username must be exactly the user's discord name (ex: Oh Snapple#2136). This is to ensure no accidental deletions.

