Project Title: PokéSight

Project Summary

PokéSight is a web-based application designed to empower Pokémon Go players by providing a smart, data-driven map of Pokémon locations. By integrating a comprehensive dataset of Pokémon statistics with a database of historical spawn locations, our app offers more than just a simple map. It allows users to perform highly specific searches, such as finding all "Legendary" rarity Pokémon with an attack stat above 250 within a five-mile radius.

The core mission of PokéSight is to transform the often random and time-consuming process of hunting for specific Pokémon into a more strategic and rewarding experience. Instead of wandering aimlessly, players can use our platform to identify spawning locations for the types of Pokémon they need, saving time and increasing their chances of success. The app will feature sophisticated filtering options by Pokémon characteristics and/or geographical sightings.

Application Description

We want to build a highly interactive and intelligent mapping tool for Pokémon Go players. The fundamental problem we're solving is the inefficiency of Pokémon hunting. Currently, players rely on luck, anecdotal information, or very basic maps. Our application will centralize static Pokémon data (types, stats, rarity) and dynamic location data (where they have been found) to create a powerful, searchable tool. Users will be able to filter the map not just by a Pokémon's name, but by its intrinsic qualities like type, rarity, and even specific base stats, allowing a trainer to find the strongest potential Pokémon in their vicinity.

Creative Component

Our creative component will be creating a heatmap visualization of Pokémon locations. It will create a visual representation using historical spawn data and the user's filters.

Here's how it'll work:

1. Data Aggregation: We'll analyze our pokemon_spawns dataset, looking at the latitude, longitude, and density of sightings of Pokémons with the user's filtered stats.
2. Environmental Correlation (API Integration): We'll integrate a third-party API, like the GoogleMap API, to fetch and display geographical data for spawn locations.

Usefulness

This application is highly useful for both casual and dedicated Pokémon Go players who want to optimize their gameplay. It adds a layer of strategy to the game, making it more engaging and less of a grind.

Basic Functions:

- View a map of your current location.
- Search for a specific Pokémon by name, stats, and range.
- Tap on a Pokémon icon to see its stats (type, rarity, base attack/defense/stamina).

Complex Functions:

- Advanced Filtering: Filter the map to show Pokémon that meet multiple criteria simultaneously (e.g., Type = 'Ghost' AND Rarity = 'Legendary').
- Stat-Based Search: The user can enter ranges for stats like maximum combat power.
- Community Reporting: Allow users to report new sightings, which are then added to the database, users can also delete or edit their reports.

Comparison to Similar Apps: While apps like Niantic's Campfire or The Silph Road provide community and information, PokéSight is different. Campfire is more of a social tool, and The Silph Road is for trading with other players. Our key differentiators are the deeply integrated stat-based filtering.

Realness

We will use two primary data sources to power our application.

1. Pokémon Stats Dataset:
    - Source: A dataset from Kaggle that the author collected from the RapidAPI Pokemon GO API link
    - Format: CSV file
    - Data Size: Cardinality of ~1,000 records (one for each unique Pokémon), with a degree of 24 attributes.
    - Information Captured: various pokemon stats like the name, base stats, rarity, type, their movesets, and their max combat power
2. Pokémon Spawn Locations Dataset:
    - Source: A publicly available dataset of Pokémon Go sightings found on Kaggle link
    - Format: CSV file
    - Data Size: Cardinality of ~300,000 historical spawn records, with a degree of 208 attributes

    - Information Captured: geographical data about where the pokemon was found, its pokedex number, the time of day the pokemon was found, weather during the day, etc.

Detailed Functionality

The application will be highly interactive, with a focus on searching and community contributions.

Functionality List:

- Guest User:
    - Search/Read: Can search for any Pokémon by name or by applying filters (type, rarity, stats). They can view all Pokémon on the map and access their detailed stats.
- Registered User:
    - Search/Read: Has all the functionality of a guest user.
    - Create: Can add (insert) a new Pokémon sighting. They can submit a form with pokemon, location, and time to our spawn database.

Low-Fidelity UI Mockup:

The main interface will be a search page for a certain pokemon with geographical preferences.

- Top Left Bar:
  - A search bar for Pokémon names.
  - Scrollable List of Pokemon with previews.
- Bottom Left Bar: Submit a Pokemon sighting Button.
- Side Panel (collapsible):
  - Filter for search (If user is unsure of a specific pokemon): Type, Combat Power, Rarity
  - Geographical Preferences (Where/when user wants to find this pokemon): Distance, Location, Weather

The interface when they have chosen their pokemon.

- Large Map View: Display where nearby pokemons are
- Pokemon stats and image on the top left of the map with its stats
- Scrollable facts about their sighting on the top right. (exact location they are seen at, time and weather)

The interface when they have chosen to report a sighting

- User is required to login to track their reports and delete a report
- Registered User can submit location, pokemon name, and time they saw a pokemon

The interface when they have chosen to delete a report

- Main box to scroll through user’s previous reports
- Large are you sure prompt so the user has a second chance

CRUD Operations:

Create (Add):

- User: Add (submit) a Pokémon sighting via Bottom Left Bar button (location, Pokémon name, time).
- User: Add filters/preferences in Side Panel (Type, CP, Rarity, Location, Distance).

Read (View):

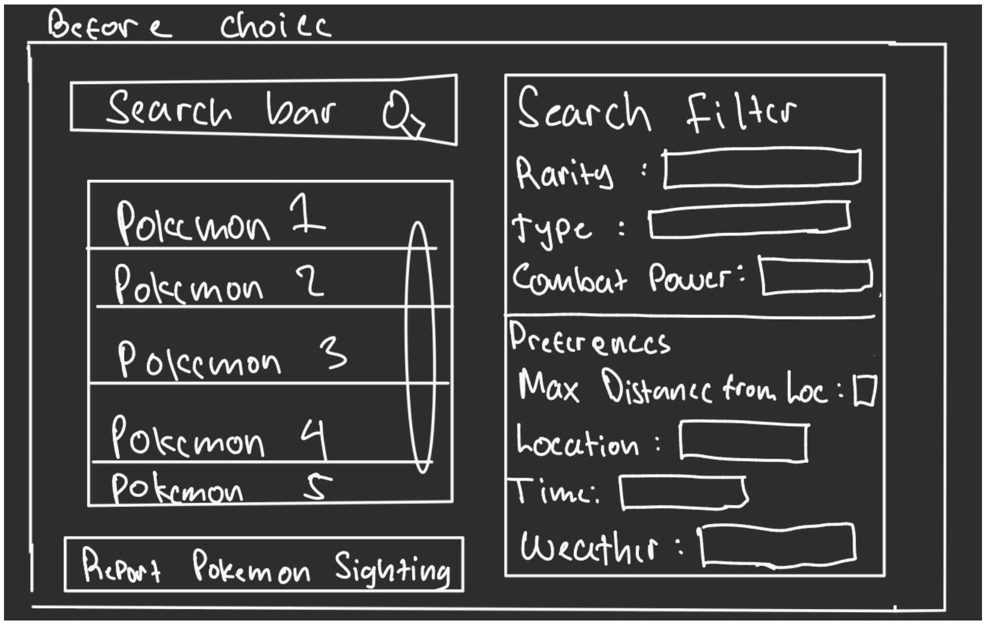
- User: Search Pokémon names in Top Left Bar search bar.
- User: View scrollable Pokémon list with previews in Top Left Bar list.
- User: View nearby Pokémon on Large Map View.
- User: View Pokémon stats and image in Top Left overlay on Map.
- User: View scrollable facts (location, time, weather) in Top Right overlay on Map.

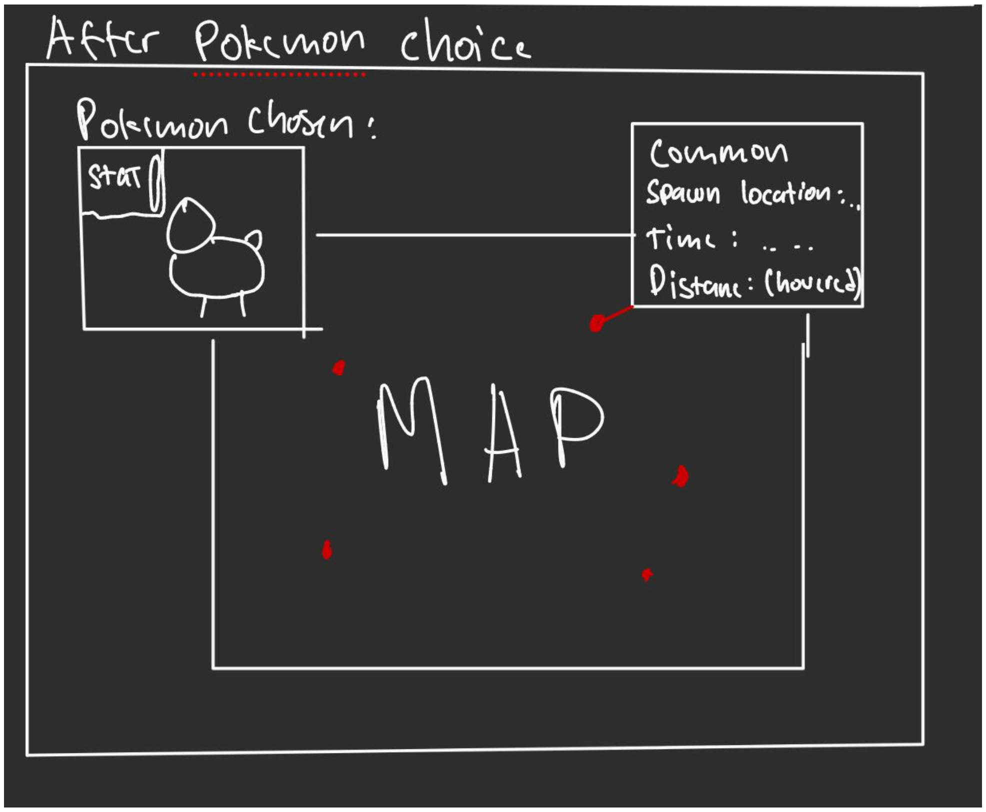
Update (Edit):

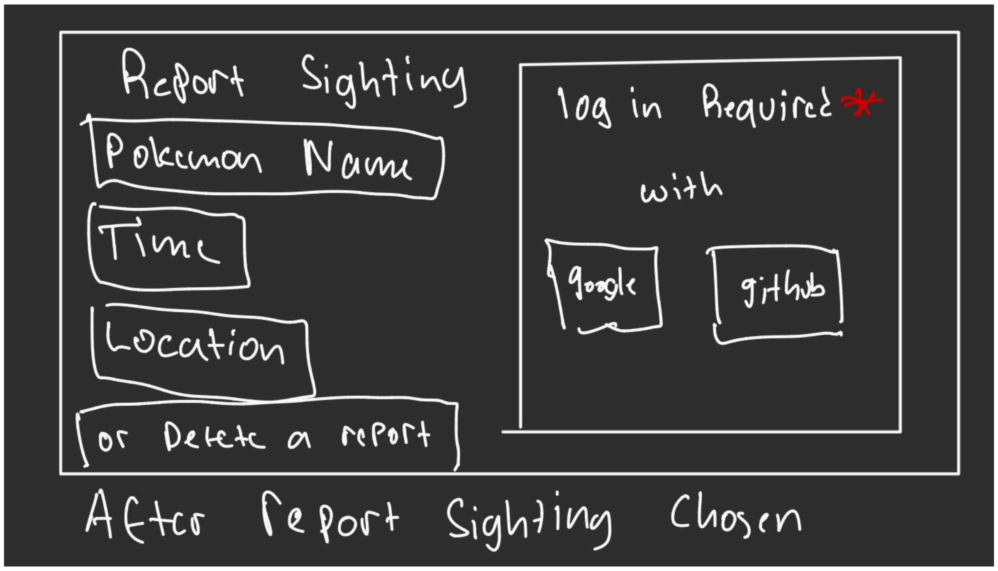
- User: Update filters/preferences in Side Panel.
- User: Update submitted sightings (edit location or time) in sighting submission interface.

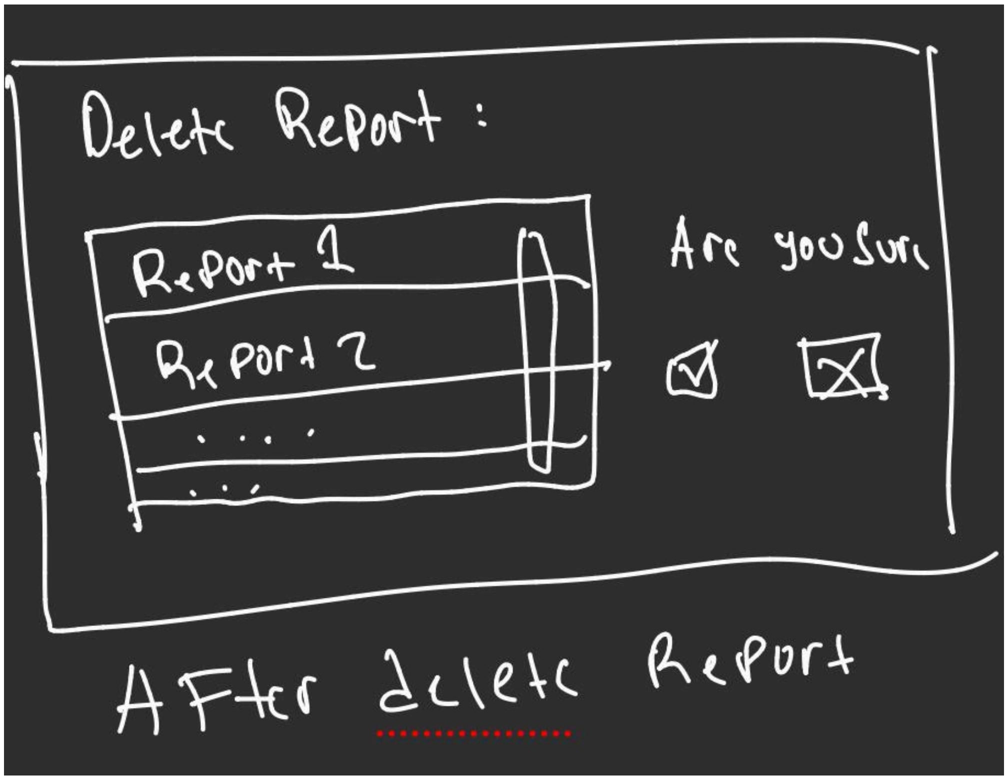
Delete (Remove):

- User: Delete their submitted sighting from Scrollable facts panel.
- User: Delete filters/preferences (reset) from Side Panel.


Before choice
Search bar
Pokemon 1
Pokemon 2
Pokemon 3
Pokemon 4
Pokemon 5
Report Pokemon Sighting
Search Filter
Rarity :
Type :
Combat Power:
Preferences
Max Distance from Loc :
Location :
Time:
Weather :


After Pokemon choice
Pokemon chosen:
STAT
Common
Spawn location: ..
Time : .. . .
Distance: (hovered)
MAP





Project Work Distribution

Task Distribution:

- Alex - Frontend
- Fino - Backend
- Avyay - Backend
- Zach - Frontend

Backend System Distribution

- **Avyay: Database Architect & Administrator** Avyay will design the entire PostgreSQL database schema and is responsible for writing the scripts to import the initial Pokémon stats and spawn location datasets. He will ensure all data relations and performance indexes are correctly established from the start.
- **Zach: User Authentication & Management** Zach is tasked with enabling all user functionality by creating the users database table. He will build the essential API endpoints for user registration and login to manage secure access to the application.
- **Fino: Core API & CRUD Operations** Fino will develop the core API endpoints using Python and FastAPI for all primary data interactions. This includes handling the searching/filtering of Pokémon and managing the CRUD operations for community-reported spawn sightings.
- **Alex: External API Integration & Geospatial Logic** Alex will handle the integration of external APIs, like weather or mapping services, to display our location data. He is responsible for creating the advanced geospatial logic and the main predictive hotspot endpoint that forecasts Pokémon locations for the frontend.