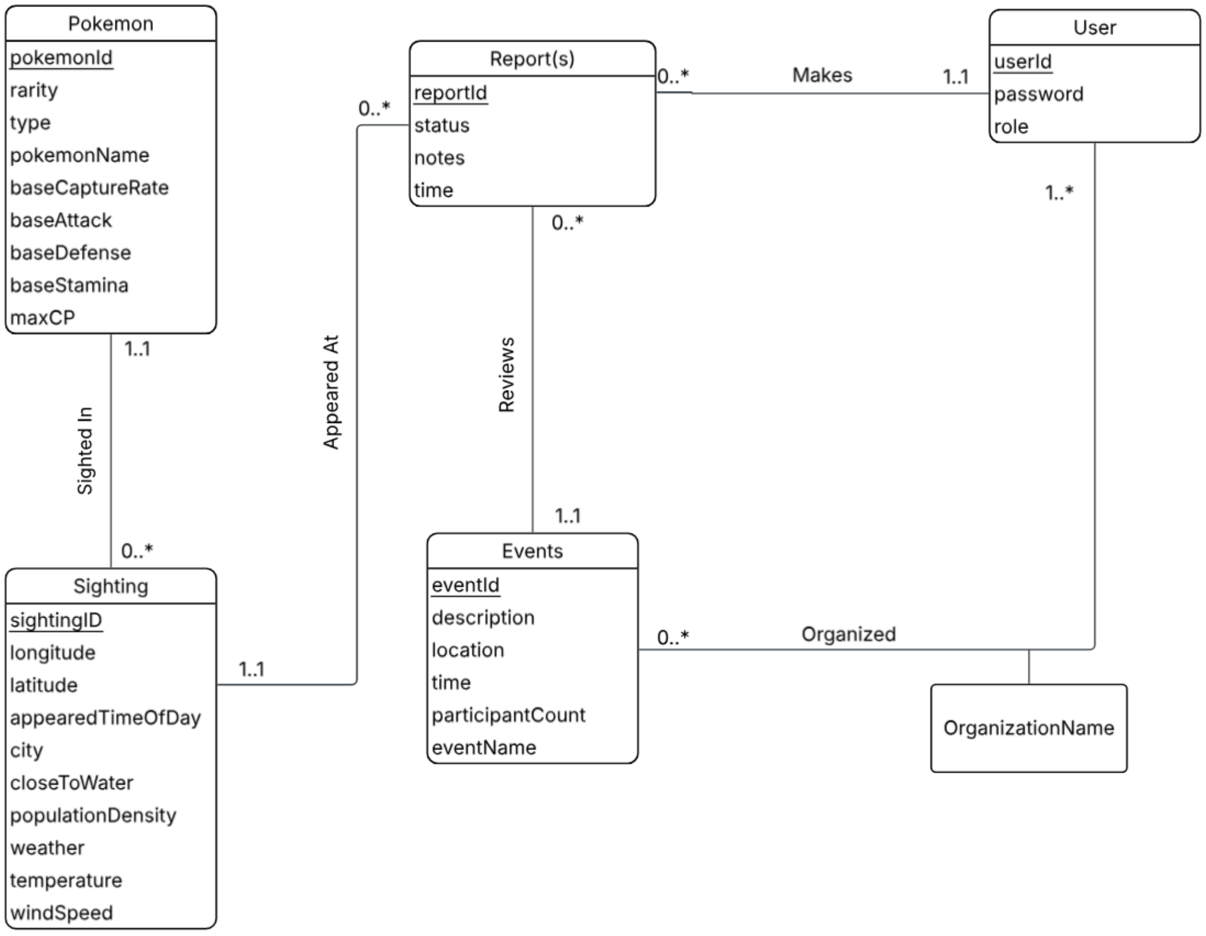
PokeSight

1 | UML Diagram



2 | Entities Descriptions

2.1 Pokémon

2.1.1 Attributes

Entity POKEMON contains pokemonId, pokemonName, rarity, type, baseCaptureRate, baseAttack, baseDefense, baseStamina, and maxCP.

2.1.2 Clarification

- Assumption: Each Pokémon has distinct properties (rarity, type, stats). These attributes and the fact that many sightings/reports can reference the same Pokémon justify treating Pokémon as a separate entity, not just an attribute.
- Modeled as Entity: Because each Pokémon (Pikachu, Eevee, etc.) can appear in lots of different sightings, and each sighting can reference a Pokémon.

2.2 User

2.2.1 Attributes

Entity USER contains userId, password, role, and organizationName.

2.2.2 Clarification

- Assumption: Users are main actors, able to create reports and events. Each user must be identifiable (by userId) , can have roles (admin, etc.), many users can be a part of the same organization.
- Modeled as Entity: Because distinct users, with credentials and roles, need their own records for authentication, authorization, and tracking contributions.

2.3 Sighting

2.3.1 Attributes

Entity SIGHTING contains sightingId, longitude, latitude, appearedTimeOfDay, weather, temperature, windSpeed, city, populationDensity, and closeToWater.

2.2.2 Clarification

- Assumption: Sighting events are discrete occurrences, each tied to a unique place and time. Each sighting must be distinguishable, capturing specific environmental and location data.
- Modeled as Entity: Because each sighting is a distinct instance with its own spatial and atmospheric characteristics, it requires an individual record. This separation allows for in-depth analysis of trends and spawn conditions.

2.4 Report

2.4.1 Attributes

Entity REPORT contains reportId, status, notes, and time.

2.2.2 Clarification

- Assumption: Reports are user-generated records that document specific pokemon sightings and reviews of events, with each report possessing its own details such as status, timestamp(when it was made), and notes (user specified). Each report must be uniquely attributable to its author.
- Modeled as Entity: Because there are 2 kinds of reports, a pokemon sighting or a review of an event. Reports also have a note attribute where users can save their own personalized notes for that specific report, with its status and time uploaded saved.

2.5 Event

2.5.1 Attributes

Entity EVENT contains eventId, eventName, description, location, time, and participantCount.

2.2.2 Clarification

- Assumption: Events are planned activities or meetups that involve users gathering at specific places and times (for trades/battles). Each event must be clearly defined with distinct details such as name, location, and scheduled time.
- Modeled as Entity: Because events are complex objects that connect multiple users (participants) to a single occurrence, each event requires a dedicated record to capture its attributes and relationships.

---

3 | Relationships Description

3.1 Pokémon Spawns

3.1.1 Cardinality

One-to-Many (One Pokémon → Many Sightings)

3.1.2 Description

A single Pokémon species can appear in multiple sightings, but each individual sighting event features only one Pokémon species. This relationship allows the system to track every unique appearance of a Pokémon while maintaining a single, authoritative record for each species' static data.

3.2 User Submits Report

3.2.1 Cardinality

One-to-Many (One User → Many Reports)

3.2.2 Description

A user can submit multiple reports over time, but each report is authored by only one user. This ensures accountability and allows for consistent tracking of individual user contributions and reliability.

3.3 Report Verifies Sighting

3.3.1 Cardinality

Many-to-One (Many Reports → One Sighting)

3.3.2 Description

Multiple users can file a report about a Pokémon sighting at the same time and location to confirm its existence and details. However, each individual report can only reference one specific sighting event. This relationship is crucial for crowdsourcing data verification.

3.4 User Creates Event

3.4.1 Cardinality

Many-to-Many (Many Users (In the same organization) → Many Events)

3.4.2 Description

Many users can be part of an organization to create and manage multiple events. This establishes clear ownership for event management while allowing active users to organize many meetups.

3.5 Users Create Reports On Events

3.5.1 Cardinality

Many-to-one (Many Reports → Review One Event)

3.5.2 Description

Users can create reports reviewing an event. This allows users to express their opinions on events to event management.

---

4 | Normalization Process

In this project, we use the Boyce-Codd Normal Form (BCNF) to normalize our database schema.

4.1 Pokémon

4.1.1 Key Analysis

Based on the functional dependencies (FD):

- pokemonId → rarity, type, pokemonName, baseCaptureRate, baseAttack, baseDefense, baseStamina
- baseAttack, baseDefense, baseStamina → maxCP

From these FDs, we can see that pokemonId is a superkey. However, the second FD shows that maxCP is determined by the base stats, which are non-key attributes in the original relation, creating a transitive dependency.

4.1.2 BCNF

The dependency baseAttack, baseDefense, baseStamina → maxCP violates BCNF because the determinant (baseAttack, baseDefense, baseStamina) is not a superkey. To resolve this, we decompose the relation into two:

- **Pokemon**: (pokemonId, rarity, type, pokemonName, baseCaptureRate, baseAttack, baseDefense, baseStamina)
- **StatsCP**: (baseAttack, baseDefense, baseStamina, maxCP)

4.2 Sighting

4.2.1 Key Analysis

Based on the functional dependencies (FD):

- sightingId → longitude, latitude, appearedTimeOfDay, weather, temperature, windSpeed, city, populationDensity, closeToWater
- longitude, latitude → city, populationDensity, closeToWater

The primary key is SightingId. However, the location attributes (city, populationDensity, closeToWater) depend on the coordinates (longitude, latitude), which is a partial dependency and violates BCNF because longitude, latitude is not a superkey.

4.2.2 BCNF

To resolve the BCNF violation, we decompose the relation into:

- **Sighting**: (sightingId, appearedTimeOfDay, weather, temperature, windSpeed, longitude, latitude)
- **Location**: (longitude, latitude, city, populationDensity, closeToWater)

4.3 User, Reports, and Events

4.3.1 Key Analysis

The User, Reports, and Events relations are already in BCNF.

- **User**: userId → password, role
- **Reports**: reportId → status, notes, time
- **Events**: eventId → description, location, time, participantCount, eventName

In each case, the only non-trivial functional dependency has the primary key as its determinant, which is a superkey. Therefore, no decomposition is needed.

---

5 | Translated Relational Schema

According to the normalization above, we can translate our ER model into the following relational schema.

5.1 Pokémon Info

This table contains static data for each Pokémon species.

```
Pokemon(
   pokemonId: INT [PK],
   pokemonName: VARCHAR(255),
   rarity: VARCHAR(255),
   type: VARCHAR(255),
   baseCaptureRate: DECIMAL,
   baseAttack: INT [FK to StatsCP.baseAttack],
   baseDefense: INT [FK to StatsCP.baseDefense],
   baseStamina: INT [FK to StatsCP.baseStamina]
)
```

This table stores the relationship between base stats and max CP.

```
StatsCP(
   baseAttack: INT [PK],
   baseDefense: INT [PK],
   baseStamina: INT [PK],
   maxCP: INT
)
```

5.2 Sighting & Location Info

This table contains static information about a geographic location.

```
Location(
   longitude: DECIMAL [PK],
   latitude: DECIMAL [PK],
   city: VARCHAR(255),
   populationDensity: INT,
   closeToWater: BOOLEAN
)
```

This table contains data for each unique Pokémon sighting event.

```
Sighting(
   sightingId: INT [PK],
   pokemonId: INT [FK to Pokemon.pokemonId],
   longitude: DECIMAL [FK to Location.longitude],
   latitude: DECIMAL [FK to Location.latitude],
   appearedTimeOfDay: VARCHAR(255),
   weather: VARCHAR(255),
   temperature: DECIMAL,
   windSpeed: DECIMAL
)
```

5.3 User Info

This table contains data for registered users.

```
User(
    userId: INT [PK],
    password: VARCHAR(255),
    role: VARCHAR(255)
    organizationName: VARCHAR(255) [FK to Organizations.organizationName]
)
```

5.4 Report Info

This table contains all user-submitted reports on sightings.

```
Reports(
    reportId: INT [PK],
    sightingId: INT [FK to Sighting.sightingId],
    userId: INT [FK to User.userId],
    eventId: INT [FK to Events.eventId],
    status: VARCHAR(255),
    notes: TEXT,
    time: DATETIME
)
```

5.5 Event Info

This table contains information on all user-created events.

```
Events(
    eventId: INT [PK],
    eventName: VARCHAR(255),
    description: TEXT,
    location: VARCHAR(255),
    time: DATETIME,
    participantCount: INT,
    organizationName: VARCHAR(255) [FK to Organizations.organizationName]
)
```

Organizations(

```
        organizationName: VARCHAR(255) [PK]
)
```