# Trail & Hiking Database using Redis

Joji Araki

## About

---

A Trail and Hiking Database that is a comprehensive resource for outdoor enthusiasts, providing detailed information about hiking trails.

## Overview

---

**Trail Information:**

- **Trail Name**
- **Location**
- **Distance**
- **Difficulty Level** (Easiest, Moderate, Moderately Strenuous, Strenuous, Very Strenuous)
- **Elevation Change**
- **Trail Type (**loop, out-and-back, or point-to-point)

**Description and Features:**

- **Trail Description**
- **Scenic Views**
- **Flora and Fauna**
- **Geological Features**

**Amenities and Facilities:**

- **Parking**
- **Restrooms**
- **Water Sources**
- **Picnic Areas**

**Trail Conditions and Alerts:**

- **Current Conditions**
- **Alerts and Closures**

**User Reviews and Ratings:**

- **User Ratings**
- **User Reviews**

**Map:**

- **Trail Map**
- **GPS Coordinates**

**Safety Information:**

- **Trailhead Signage**
- **Emergency Contacts**
- **Wildlife Safety**

**User Contributions:**

- **Photo Gallery**
- **Trail Logs**

## Rules

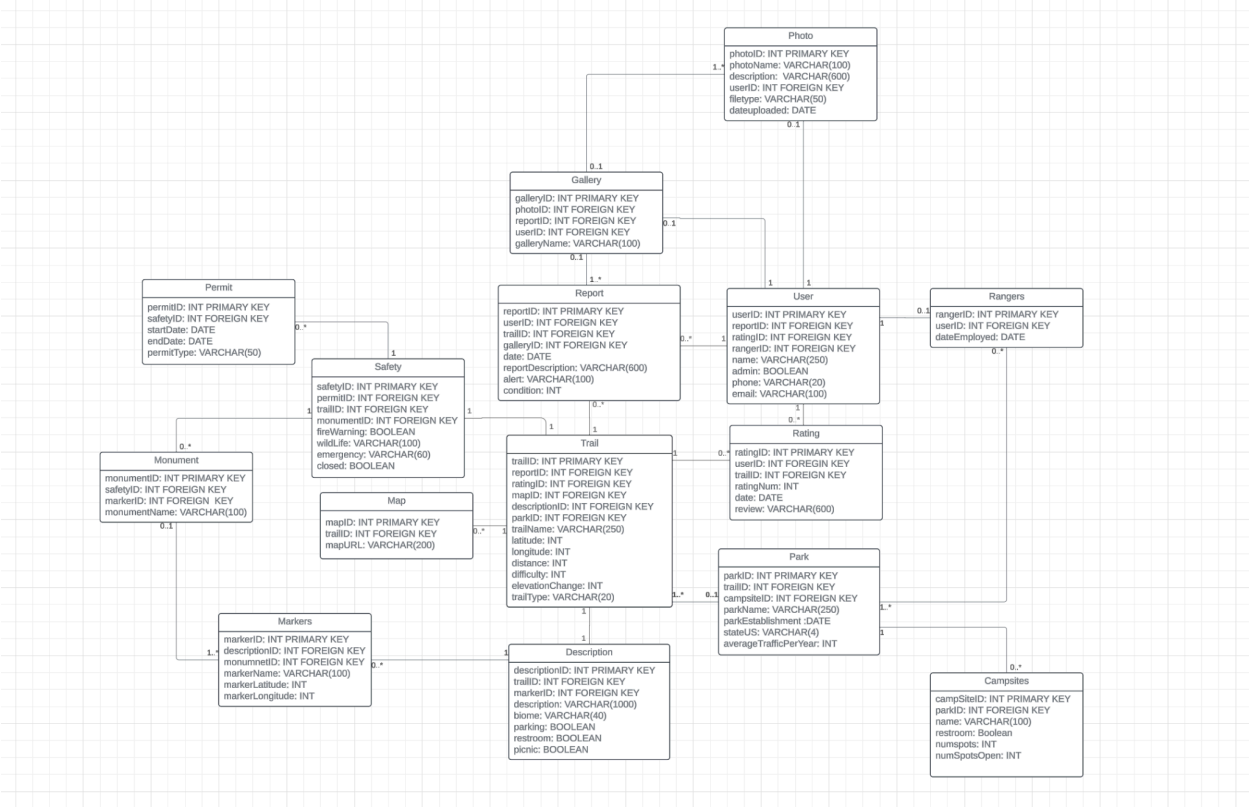
---

Nouns | Verbs

- Trail information which will have the name of the trail, location, distance, difficulty level based on elevation gain and distance, and a trail type
- Each trail will have an identification id and connected to it will be a description and features of the trail
- Connected to the description and features will be information about amenities and facilities such as bathroom, water, parking and picnic
- There will also be trail conditions and alerts that are updated in real time in order for hikers to stay safe
- User reviews with both user information, review, and rating associated with each
- Users able to write and submit reports of safety or about trail
- Safety information provided for each trail can be linked to condition and alerts
- Add and edit trails
- Map link and relative GPS coordinate area
- Trails must have name, location, distance, difficulty and type
- Users can update trail condition with data associated
- Safety alerts only by admin accounts
- GPS start and end points

- In memory key value database can be used to view most visited trails
- Can be utilized to get the safety and the risk of the trail prior to other information
- Connect to the database so that location is transferred between back and forth so that in the case that the hiker is lost, we have some sort of information to look back to

# UML Diagram




# Implementing Functionalities

1. **Trail Information**:

**Redis Hash**: We can use a hash to store information about each trail, including its name, location, distance, difficulty, and type. Example we can do is:
HSET trail_information:Trail1 name “Trail 1”

2. **Description and Features**:

**Redis Hash**: We can create a hash for each trail's description and features, with the trail ID as the key. Example we can do is:
HSET trail_description: Trail1 description “Text”

3. **Amenities and Facilities**:

**Redis Hash**: We can store information about amenities and facilities for each trail using a hash structure. Example we can do is:
HSET trail_facilities:Trail1 parking "Open"

4. **Trail Conditions and Alerts**:

**Redis Hash**: We can store real-time updates on trail conditions and alerts using a hash structure. Example we can do is:
HSET trail_safety:Trail1 time "Sunny, 40*F"

5. **User Reviews**:

**Redis Hash**: We can store user reviews for each trail using a hash structure, with the user ID as the key. Example we can do is:
HSET trail_review:Trail1:userID review "Text"

6. **User Reports**:

**Redis List**: We can use a list to store user reports for each trail, with each report being a new entry in the list. Example we can do is:
LPUSH trail_report:Trail1 "Warning: Text"

7. **Safety Information**:

**Linking Other Data**: Safety information can be linked to trail conditions and alerts using references or IDs, ensuring that users can access safety information relevant to the current trail conditions.

8. **GPS Coordinates**:

**Redis Geospatial Index**: We can store GPS coordinates for the start and end points of each trail using a geospatial index. Example we can do is:
GEOADD trail_cord: -43 23 "Trail1_begins"

10. To view most visited Trails

Redis Sorted Set: We can use a sorted set to store the most visited trails. Each trail will be a member of the sorted set, and the score will represent the number of visits or views. We can apply this by using key's such as "mostVisited:trailID". userID as the values and the score is for the number of visits the trail has had. Whether that be from different individuals or repeat visits.

11. Getting risk and safety first

**Redis Hash**: We can use a hash to store safety and risk assessment information for each trail. Each trail ID can be the key, and the hash fields can represent various safety metrics or risk factors. Key trail_safety: Trail1 represents the

hash for TrailA, and the fields represent safety metrics such as weather conditions, trail condition, and wildlife presence.

12. Hiker Coordinate

**Redis Geospatial Index**: We can use Redis' Geospatial data structures to store individuals locations.In this example, the key user_locations represents the geospatial index, where each trail is stored with its longitude and latitude coordinates. This can be useful for making sure that individuals are safe on on the trail.

## Redis Commands

---

1. **Trail Information**:

**Create/Update Trail Information**
HSET trail_information:Trail1 name "Trail 1"
**Read Trail Information**:
HGET trail_information:Trail1 name
**Delete Trail Information**:
DEL trail_information:Trail1

2. **Description and Features**:

**Create/Update Description and Features**:
HSET trail_description:Trail1 description "Text"
**Read Description and Features**:
HGET trail_description:Trail1 description
**Delete Description and Features**:
DEL trail_description:Trail1

3. **Amenities and Facilities**:

**Create/Update Amenities and Facilities**:
HSET trail_facilities:Trail1 parking "Open"
**Read Amenities and Facilities**:
HGET trail_facilities:Trail1 parking
**Delete Amenities and Facilities**:
DEL trail_facilities:Trail1

4. **Trail Conditions and Alerts**:

**Create/Update Trail Conditions and Alerts**:
HSET trail_safety:Trail1 time "Sunny, 40°F"
**Read Trail Conditions and Alerts**:

HGET trail_safety:Trail1 time
**Delete Trail Conditions and Alerts**:
DEL trail_safety:Trail1

5. **User Reviews**:

**Create/Update User Reviews**:
HSET trail_review:Trail1:userID review "Text"
**Read User Reviews**:
HGET trail_review:Trail1:userID review
**Delete User Reviews**:
DEL trail_review:Trail1:userID

6. **User Reports**:

**Create User Reports**:
LPUSH trail_report:Trail1 "Warning: Text"
**Read User Reports**:
LRANGE trail_report:Trail1 0 -1
**Delete User Reports**:
DEL trail_report:Trail1

7. **Safety Information**:

**Create/Update Safety Information**:
HSET trail_safety:Trail1 weather_conditions "Sunny" HSET trail_safety:Trail1 trail_condition "Good" HSET trail_safety:Trail1 wildlife_presence "Low"
**Read Safety Information**:
HGETALL trail_safety:Trail1
**Delete Safety Information**:
DEL trail_safety:Trail1

8. **GPS Coordinates**:

**Create/Update GPS Coordinates**:
GEOADD trail_cord -43 23 "Trail1_begins"
**Read GPS Coordinates**:
GEOPOS trail_cord "Trail1_begins"
**Delete GPS Coordinates**:
ZREM trail_cord "Trail1_begins"

9. **Most Visited Trails**:

**Increment Trail Visits**:
ZINCRBY mostVisited:trailID 1 "UserID"

**Retrieve Most Visited Trails**:
ZREVRANGE mostVisited:trailID 0 -1
**Delete Most Visited Trails**:
DEL mostVisited:trailID

To Retrieve the Most Visited Trail
ZINCRBY topVisited:trailID 1 “UserID”
ZREVRANGE topVisited:trailID 0 -1

10. **Getting Risk and Safety First**:

**Create/Update Safety and Risk Information**:
HSET trail_safety:Trail1 weather_conditions "Sunny" HSET trail_safety:Trail1 trail_condition "Good" HSET trail_safety:Trail1 wildlife_presence "Low"
**Read Safety and Risk Information**:
HGETALL trail_safety:Trail1
**Delete Safety and Risk Information**:
DEL trail_safety:Trail1

11. **Hiker Coordinate**:

**Create/Update Hiker Coordinates**:
GEOADD user_locations -43 23 "hiker1"
**Read Hiker Coordinates**:
GEOPOS user_locations "hiker1"
**Delete Hiker Coordinates**:
ZREM user_locations "hiker1"