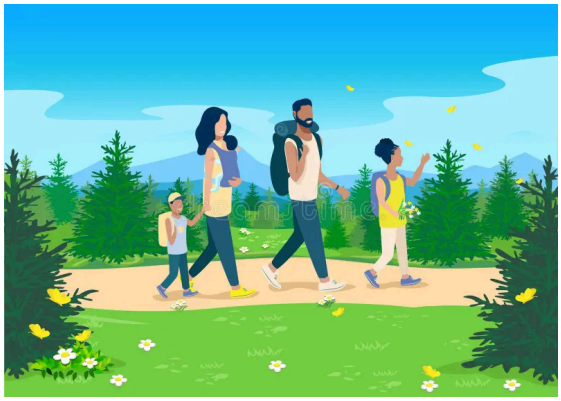
Scenario: Hiking Trails Management Platform

Overview:

You are tasked with developing a backend system for a platform that allows outdoor enthusiasts to discover and share hiking trails. Users can explore different trails, view trail details such as difficulty level, length, and elevation gain, and share their own experiences.



Key Features:

1. **Trail Management:**
    - Users can browse a list of available hiking trails.
    - Trail details include name, location, difficulty level, length, elevation gain, and description.
2. **Review and Rating:**
    - Users can view reviews and ratings for each trail.
    - They can submit their own reviews along with a rating.
3. **Search and Filter:**
    - Users can search trails by location, difficulty level, or length.
    - They can filter trails by minimum and maximum difficulty levels, length, and elevation gain.

Schema and Validations:

Field	Validation
trailId	Unique identifier (automatically generated)
name	Required, String
location	Required, String
difficulty	Required, String (easy, moderate, difficult)
length	Required, Number (positive value)
elevationGain	Required, Number (non-negative value)
description	String
reviews	Array of Objects
- rating	Number (1 to 5)
- comment	String

Routes:

1. **Basic CRUD Routes:**
    - GET /trails: Get all hiking trails
    - GET /trails/:trailId: Get details of a specific trail
    - POST /trails: Add a new trail

- PUT /trails/:trailId: Update details of a trail
- DELETE /trails/:trailId: Remove a trail

2. **Review and Rating Routes:**
    - POST /trails/:trailId/reviews: Add a review for a trail
    - PUT /trails/:trailId/reviews/:reviewId: Update a review for a trail
    - DELETE /trails/:trailId/reviews/:reviewId: Remove a review for a trail
3. **Search and Filter Routes:**
    - GET /trails/search?location=: Search trails by location
    - GET /trails/filter?difficulty=: Filter trails by difficulty level
    - GET /trails/filter?minLength=: Filter trails by minimum length
    - GET /trails/filter?maxLength=: Filter trails by maximum length
    - GET /trails/filter?minElevation=: Filter trails by minimum elevation gain
    - GET /trails/filter?maxElevation=: Filter trails by maximum elevation gain

Additional Features:

- Get top-rated trails based on user ratings.
- Allow users to sort trails by rating, length, or elevation gain.
- Calculate average rating for each trail based on user reviews.
- Provide statistics such as total number of trails, average length, and average elevation gain.

Technical Implementation:

1. Node.js and Express.js:
    - Use Node.js for server-side logic.
    - Utilize Express.js to create RESTful APIs for managing trails, reviews, and search/filter functionalities.
2. Mongoose:
    - Define Mongoose schemas for trails and reviews with the specified fields and validations.

- Implement CRUD operations and search/filter functionalities using Mongoose methods.

3. Routes Implementation:
    - Define route handlers for each endpoint, implementing the necessary logic to handle requests and responses.
    - Utilize MongoDB queries to perform search, filter, and aggregation operations.