**Job Recommendation Service Documentation**

1. Overview
2. Recommendation Logic
3. Assumptions
4. Design Decisions
5. Challenges and Solutions
6. Conclusion

**Overview**

This document outlines the design and implementation of the job recommendation service, detailing the recommendation logic, assumptions made, and challenges encountered during development.

**Recommendation Logic**

Matching Algorithm

The recommendation algorithm aims to match user profiles with job postings based on several criteria:

1. **Skills Matching**:

The algorithm checks if the user's skills overlap with the required skills for job postings. A match is considered successful if there is at least one common skill.

2. **Experience Level**:

The user's experience level must meet or exceed the job posting's required experience level. For instance, if a job posting requires an "Intermediate" experience level, it will be suitable for users classified as "Intermediate" or "Senior."

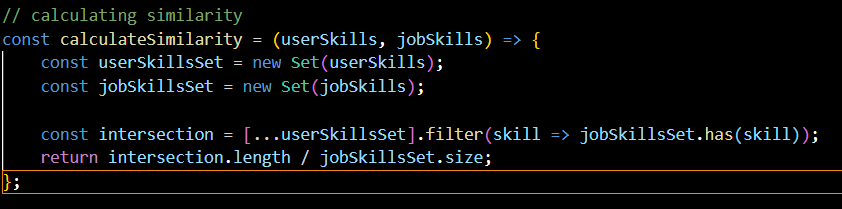
3. **Location Preferences**:

The algorithm filters job postings based on the user's location preferences. A job is included in the recommendations if the job location matches any of the user's preferred locations.

4. **Job Type**:

The job postings must match the user's desired job type (e.g., Full-Time, Part-Time). If the user's preference is for "Full-Time," only full-time job postings will be considered.

**Similarity Calculation**



User Skills: ["JavaScript", "Node.js"]

Job Skills: ["JavaScript", "React"]

Common Skills (Intersection): ["JavaScript"]

Score Calculation:

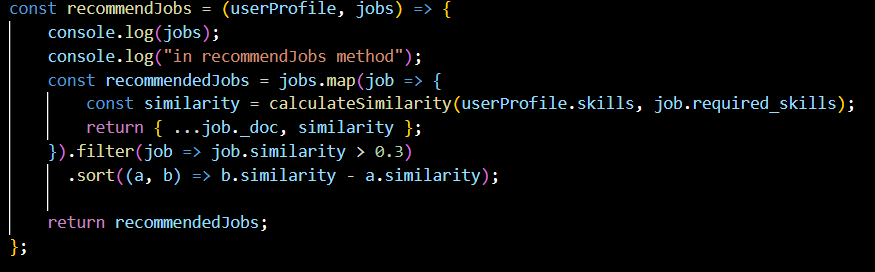
intersection.length = 1 (only "JavaScript")

jobSkillsSet.size = 2 (skills required for the job: "JavaScript" and "React")

So, the similarity score would be 1 / 2 = 0.5.

**Recommendation Function**

This function generates recommendations based on the similarity scores.



Input: userProfile (contains user skills, experience, preferences) and jobs (list of jobs fetched from the database).

Output: An array of recommended job postings, sorted by their similarity scores.

**Assumptions**

**Skill Importance**: Skills are treated as the primary factor in matching, assuming that having relevant skills significantly increases the likelihood of job success.

**Experience Levels**: The experience level is categorized into predefined tiers (e.g., Junior, Intermediate, Senior), and each job has a strict requirement.

**Location and Job Type**: The user's preferences for job location and type are assumed to be flexible, where a match can lead to consideration for jobs in adjacent locations (e.g., Remote jobs).

**Design Decisions**

Database Choice: MongoDB was chosen for its flexibility in storing semi-structured data and ease of scaling as the application grows.

RESTful API Design: The API adheres to RESTful principles, providing clear and logical endpoints for user interactions.

**Challenges and Solutions**

Finding effective matching logic has been challenging, especially with the diverse factors that influence job recommendations.

Each job posting is evaluated against the user profile to generate a score. The scoring is calculated as follows:

Skills Match (50%): A comparison is made between the user’s listed skills and the required skills for each job posting. Each matching skill contributes to the overall score.

Job Type Match (10%): The algorithm checks if the job type (e.g., Full-Time, Part-Time) aligns with the user’s preferences, contributing to the score if there is a match.

**Conclusion**

The job recommendation service successfully matches users with job postings based on skills, experience, location, and job type. Future enhancements may include machine learning techniques for better predictions based on user interactions and feedback.