Determining User Proficiency through Project Contributions

1. Code Contributions (Pushes)

- How to Implement:

  - Data Collection: Use Git APIs (e.g., GitHub, GitLab) to fetch commits by user and track the languages used.

  - Code Language Detection: Use tools like GitHub Linguist or custom scripts to identify languages from file extensions.

  - Weighting: Assign weight to user proficiency based on their language usage, e.g., Python, JavaScript, SQL.

- Tools to Use: GitHub API, GitLab API, Bitbucket API.

2. Pull Request Creation

- How to Implement:

  - Data Collection: Track how many pull requests (PRs) a user creates via the Git platform APIs.

  - Weighting: Higher weight for PRs related to feature development, using metadata like labels or branch naming conventions.

- Tools to Use: GitHub API, GitLab API, Bitbucket API.

3. Pull Request Merges

- How to Implement:

  - Data Collection: Track the PRs that are successfully merged.

  - Weighting: Assign higher weight to merged PRs since they indicate successful contributions.

- Tools to Use: GitHub API, GitLab API, Bitbucket API.

4. Pull Request Rejections

- How to Implement:
  - Data Collection: Track PRs that were closed without merging.
  - Weighting: Frequent rejections may lower proficiency ratings, especially for poor code quality.
- Tools to Use: GitHub API, GitLab API, Bitbucket API.

5. Type of Contribution

- How to Implement:
  - Data Collection: Categorize PRs and commits by type (new features, bug fixes, refactoring).
  - Weighting: Assign higher weight to complex contributions like new features or refactoring.
- Tools to Use: GitHub, GitLab APIs, and branch naming conventions.

6. Code Review Participation

- How to Implement:
  - Data Collection: Track how many code reviews a user participates in.
  - Weighting: Users providing high-quality feedback during reviews should get higher proficiency scores.
- Tools to Use: GitHub API, GitLab API.

7. Language Breakdown

- How to Implement:
  - Data Collection: Analyze file extensions in commits to detect programming languages.
  - Weighting: Weight user proficiency based on the languages they contribute to most.
- Tools to Use: GitHub Linguist, custom language detection scripts.

8. Issue Resolution and Commit Linkage

- How to Implement:
    - Data Collection: Track the issues a user resolves via commits or PRs, linked to issue trackers (e.g., Jira).
    - Weighting: Resolving complex or high-priority issues increases proficiency.
- Tools to Use: Jira API, GitHub Issues API.

9. Task Complexity

- How to Implement:
    - Data Collection: Track task complexity in project management tools like Jira.
    - Weighting: Assign higher weight for more complex tasks completed by users.
- Tools to Use: Jira API, project management tools.