Getting Started

Download necessary libraries and tools:

Run `npm install –force`

Deploy Application

Run `npm run deploy` to get the deployed app.

Deployed App URL: MoCode

BackEnd Functions

EXPORTED FUNCTIONS

isUsernameValid(username)

- Purpose:
  - Attempts to verify the validity of a LeetCode username by fetching recent submission data.
- Inputs:
  - username (string): The LeetCode username to verify.
- Outputs:
  - Returns recent submissions object if the username is valid and the API call succeeds, or False otherwise.
- Details:
  - Calls an external API to fetch recent submissions for the given username.
  - Validates the username based on the API response.
  - **Currently non-operational due to an external API dependency being down as of 3/11/24.**

populateNewUserHistory(userId, submissions)

- Purpose:
  - Integrates a user's LeetCode submission history into their profile on the platform.
- Inputs:
  - userId (string): The unique identifier of the user.
  - submissions (object): A collection of submission objects. (from previous function)
- Outputs:
  - None. Updates the user's history directly in the database.

- Details:
    - Iterates over submissions, matching each with a corresponding question in the local database and recording its completion status.
    - **Currently non-operational due to an external API dependency being down as of 3/11/24.**

generateQuestions(userData, userProblems)

- Purpose:
    - Generates and assigns a new set of problems for the user, updating their profile with the recommendations.
- Inputs:
    - userData (object): Contains information about the user.
    - userProblems (object): The current set of problems associated with the user.
- Outputs:
    - None directly. Updates the user's profile with new problem recommendations.
- Details:
    - Flushes previously recommended but unattempted problems, generates a new set of recommended problems, and updates the user's profile.

HELPER FUNCTIONS

fetchQuestions()

- Purpose:
    - Retrieves a list of coding problems.
- Inputs:
    - None.
- Outputs:
    - Returns an array of problem objects loaded from a local JSON file.
- Details:
    - Used to load the entire set of coding problems for recommendations.

FIREBASE FUNCTIONS

addUserProblemEntry(userId, question, timeStamp, status, timeDuration)

- Purpose:
    - Adds a user problem entry to Firebase.
- Inputs:
    - userId (string),
    - question (object),
    - timeStamp (Timestamp),
    - status (string),
    - timeDuration (int).

- Outputs:
    - Returns the document ID of the added or updated problem entry.
- Details:
    - Checks if an entry already exists for the given user and problem, updating or creating as necessary.

flushPreviousQuestions(userData, userProblems)
- Purpose:
    - Flushes previously recommended but unattempted questions for a user.
- Inputs:
    - userData (object),
    - userProblems (object).
- Outputs:
    - An array of deleted user problem objects.
- Details:
    - Iterates over the user's recommended problems, deleting unattempted ones from the Firebase database.

updateUserRecommendedArray(userId, problems)
- Purpose:
    - Updates the recommended problems array for a user in the database.
- Inputs:
    - userId (string),
    - problems (array).
- Outputs:
    - None. Directly updates the user's document with the new recommended problems.
- Details:
    - Responsible for updating the list of recommended problems associated with a user in the Firebase database.

REC PROBLEMS FUNCTIONS

weightedRandomSelect(problems, count, allProblems)
- Purpose:
    - Selects a specified number of problems randomly, weighted by their normalized weights.
- Inputs:
    - problems (array),
    - count (number),
    - allProblems (array).
- Outputs:
    - A subset of 'allProblems', selected based on weighted probability.

- Details:
    - Utilizes a weighted random selection algorithm to pick unique problems from the complete set.

generateProblems(userProblems, count)
- Purpose:
    - Generates a new set of problems for the user, based on past interactions and preferences.
- Inputs:
    - userProblems (array),
    - count (number).
- Outputs:
    - A list of newly selected problem objects.
- Details:
    - Generates a tailored set of problems for the user by analyzing their past problem interactions.

calculateProbability(score, k)
- Purpose:
    - Calculates the probability of selecting a problem based on its score.
- Inputs:
    - score (number),
    - k (number).
- Outputs:
    - A probability value between 0 and 1.
- Details:
    - Applies the logistic function to transform a problem's score into a probability.

calculateScoreRepeat(userProblem)
- Purpose:
    - Calculates a score for a problem based on its recurrence in the user's problem-solving history.
- Inputs:
    - userProblem (object): A specific problem object from the user's history.
- Outputs:
    - A score ranging from 0.75 to 2, reflecting the problem's relevance based on its recency and frequency of attempts.
- Details:
    - Evaluates the significance of a problem for the user by examining how recently and how frequently it has been attempted, combining recency and frequency scores.

calculateScoreNew(problem, averageDifficulty, recentCategories)
- Purpose:

    - Calculates a suitability score for a new problem based on the user's recent problem-solving history.
- Inputs:
    - problem (object): The new problem to be evaluated.
    - averageDifficulty (number): The user's recent average problem difficulty.
    - recentCategories (array): Categories of recently attempted problems by the user.
- Outputs:
    - A score (0.25 to 5) indicating the problem's suitability for recommendation.
- Details:
    - Scores problems higher if they match the user's recent categories or closely align with their average difficulty level, promoting a balanced learning experience.

recentStatistics(userProblems, problems)

- Purpose:
    - Calculates the average difficulty and identifies the recent categories of problems the user has completed.
- Inputs:
    - userProblems (array): The user's problem-solving history.
    - problems (array): The complete list of available problems.
- Outputs:
    - An array containing the average difficulty and a list of recent problem categories.
- Details:
    - Analyzes the last 10 completed problems to determine the user's recent focus areas and difficulty level, including "fake" entries for users with fewer than 10 recent completions.