**Student Project Review Management - Project Title Search Feature**

This document outlines the workflow and requirements for implementing the **Project Title Search** feature within the student dashboard. The goal is to allow a user to search for a project title, find the closest match from the database, and display the result with a similarity score.

**1. Workflow Description**

The search feature is a full-stack process that involves communication between the frontend, a Node.js Express backend, and a MongoDB database. The core functionality is to compare a user's search query with existing project titles by removing common words and calculating a similarity percentage.

1. **Frontend Action (script.js):**
   * A user enters a project title into the search box on the index.html page.
   * Upon clicking the "Search" button, the frontend logic in script.js captures the user's input.
   * It then makes an **asynchronous request** (using fetch or axios) to the backend server (server.js), sending the search query.
2. **Backend Processing (server.js):**
   * The backend server receives the search query from the frontend.
   * It establishes a connection to the **MongoDB database** (Login database, project\_title collection).
   * It fetches **all existing project titles** from the project\_title collection.
   * **Stop Word Removal:** Both the user's search query and each project title from the database are pre-processed to remove common stop words (e.g., "a", "an", "the", "is", "of", "and", "using", "for").
   * **Similarity Calculation:** A similarity algorithm is applied to compare the cleaned user query against each cleaned project title. A simple **word overlap** or **cosine similarity** algorithm is recommended to determine the percentage of similarity.
   * The backend identifies the project title with the highest similarity score.
3. **Returning the Results (server.js to script.js):**
   * The backend formats the results, including the closest matching project\_title and its calculated similarity percentage.
   * This data is sent back to the frontend as a JSON response.
4. **Frontend Display (script.js):**
   * The script.js file receives the JSON response from the backend.
   * It dynamically updates the index.html page to display the closest matching project title and its similarity percentage directly beneath the search box.

**2. Flowchart**

This flowchart visually represents the end-to-end process described above.

┌───────────────────────┐

│ User Enters Title │

└─────────┬─────────────┘

│

▼

┌──────────────────────────┐

│ Frontend Sends Fetch Req │

│ to Backend (/search) │

└─────────┬────────────────┘

│

▼

┌───────────────────────────────┐

│ Backend Receives Request │

│ (Express Route) │

└─────────┬────────────────────┘

│

▼

┌─────────────────────────────────────┐

│ Connect to MongoDB (Login DB) │

│ Get All Project Titles │

└─────────┬──────────────────────────┘

│

▼

┌─────────────────────────────────────────────┐

│ Remove Stop Words from Input + DB Titles │

└─────────┬────────────────────────────────────┘

│

▼

┌────────────────────────────────────┐

│ Calculate Similarity (word match) │

└─────────┬──────────────────────────┘

│

▼

┌───────────────────────────────────────┐

│ Return Closest Matches + Percentages │

└─────────┬─────────────────────────────┘

│

▼

┌────────────────────────────────────────┐

│ Frontend Displays Title(s) + Scores │

└────────────────────────────────────────┘