PRODUCT NAME MAPPING SYSTEM

Overview

The **Product Name Mapping System** is designed to standardize product names from different suppliers to facilitate consistent data entry. The system provides:

- **Manual Matching**: Add mappings for product names manually.
- **Automatic Matching**: Identify and map product names automatically using intelligent matching algorithms.
- **Fallback Mechanism**: Ensures the system works even if the backend server is unavailable.
- **CRUD Operations**: Create, Read, Update, and Delete mappings seamlessly.

Features and Functionalities

1. Manual Matching

- Users can manually input a supplier product name and map it to a standardized name.
- Data is updated dynamically in the UI and saved to the backend server.

2. Automatic Matching

- Intelligent matching using:
  - **Token Matching**: Splits product names into tokens (words) and compares them.
  - **Fuzzy Matching**: Calculates similarity between strings using Levenshtein distance.
  - **Synonym Handling**: Replaces common abbreviations and synonyms with standardized terms.

3. Fallback Mechanism

- Uses a predefined local dictionary if the backend server is unreachable.
- Ensures uninterrupted operation by loading and saving mappings locally.

4. CRUD Operations

- **Create**: Add new mappings manually.
- **Read**: Fetch mappings from the server or fallback dictionary.
- **Update**: Modify mappings dynamically.
- **Delete**: Remove mappings via a delete button in the UI.

Technical Details

Frontend

Languages and Libraries

- **HTML**: Structure of the interface.
- **CSS**: Styling, including flexbox for layout adjustments.

- **JavaScript**: Core functionality, including:
    - Fetch API for server communication.
    - DOM manipulation for UI updates.

Key Features in JavaScript

1. **Normalization**
    - Converts text to lowercase.
    - Removes special characters and trims spaces.
2. **Token and Fuzzy Matching**
    - **Token Matching**: Breaks product names into words and compares sets of tokens.
    - **Fuzzy Matching**: Uses Levenshtein distance to identify similar strings.

Backend

Technologies Used

- **Node.js and Express**: Server for handling API requests.
- **MongoDB**: Database for storing mappings persistently.
- **Mongoose**: ORM for interacting with MongoDB.

Note:

This version of the project uses MongoDB Compass at a local level. The API routes related to this system are also locally generated, and the data is stored locally. Additionally, the local mapping is only implemented for a subset of product names as defined in the fallback dictionary.

Cases Identified and Handled

1. Case Sensitivity

- Normalizes text to lowercase for consistent matching.

2. Extra Spaces

- Trims leading and trailing spaces and replaces multiple spaces with a single space.

3. Abbreviations and Synonyms

- Uses a dictionary to replace common abbreviations (e.g., sh → sheet).

4. Server Downtime

- Falls back to a local dictionary stored in JavaScript.

5. Exact and Partial Matches

- Handles both exact matches and approximate matches (e.g., "a4sheet" matches "a4 sheet").

How to Use

1. Adding a Mapping

- Enter the supplier product name and standardized name in the input fields.
- Click **Submit**.
- The new mapping appears in the list and is saved to the server.

2. Deleting a Mapping

- Click the **Delete** button next to a mapping.
- The mapping is removed from the list and deleted from the server.

3. Automatic Matching

- Enter a product name in the search field.
- The system uses intelligent matching algorithms to suggest a standardized name.

Future Improvements

- Add user authentication for secure access.
- Implement a frontend interface for bulk uploads.
- Use machine learning models for improved matching accuracy.
- Optimize for large datasets with pagination and caching.

Designed and Developed by

Dhanush C
Acharya Institute of Technology,
Bengaluru
dhanushchandru28@gmail.com
9901662554