**Working Prototype**

The project includes a product mapping system with the following features:

1. Frontend:
   * User-friendly interface for matching product names and adding new mappings.
   * Responsive design for mobile and desktop devices.
   * Displays current mappings dynamically for easier management.
   * Input validation and error feedback for user actions.
2. Backend:
   * REST API to handle product matching (/find), adding new mappings (/add), and retrieving current mappings (/mappings).
   * SQLite database to store standardized names and their variations.
   * Basic string normalization for handling variations like case differences, spacing, and abbreviations.
3. Database:
   * Stores mappings between product name variations and their corresponding standardized names.
   * Supports dynamic updates via user interaction.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Assumptions Made**

1. Variation Handling:
   * Minor differences in product names (case, spacing, common abbreviations) can be normalized using basic preprocessing techniques.
   * Exact match is expected after normalization for mapping success.
2. Standardized Names:
   * Users will provide a clear and unique standardized name for each product type during the mapping process.
3. Input Consistency:
   * Users will not input completely irrelevant or misspelled data. Edge cases are handled only to a basic extent.
4. API Interactions:
   * Assumes that the backend server is hosted locally on http://localhost:3000.

**Use Cases the System Can Handle**

1. Exact and Normalized Matching:
   * Can match products with different cases (A4 Paper, a4 paper) or slight differences in spacing (500sh, 500 sh).
2. Dynamic Mapping Updates:
   * Users can add new variations of product names and update the mapping dictionary dynamically.
3. Real-Time Feedback:
   * Provides immediate feedback if a match is found or prompts users to add new mappings if none exist.
4. Mapping Retrieval:
   * Displays all current mappings for transparency and easier manual updates.

**Use Cases the System Cannot Handle**

1. Complex Variations:
   * Fails to match products with entirely different phrasing or descriptions that require advanced semantic understanding.  
     Example: *"Printer Paper"* vs. *"A4 Copy Paper 500 sheets"*.
2. Domain-Specific Knowledge:
   * Cannot infer relationships between products that require knowledge of synonyms or context (e.g., *"Sticky Notes"* vs. *"Post-it Notes"*).
3. Multi-Language Support:
   * Assumes all input data is in a single language (e.g., English). Fails for multilingual variations.
4. Duplicate Variations:
   * If users unknowingly input duplicate mappings, there is no deduplication mechanism.

**Future Improvement Ideas**

1. Advanced Matching:
   * Implement a fuzzy matching algorithm (e.g., Levenshtein distance) to handle typos and more diverse variations.
   * Use Natural Language Processing (NLP) for semantic matching.
2. Duplicate Prevention:
   * Add a feature to detect and prevent duplicate entries in the mapping dictionary.
3. Bulk Import/Export:
   * Allow users to upload or download mappings in bulk (e.g., via CSV files) for better scalability.
4. Search Feature:
   * Add a search bar in the frontend to filter current mappings for easier navigation.
5. Multi-Language Support:
   * Extend the system to handle multilingual inputs using language translation APIs.
6. Analytics Dashboard:
   * Provide insights on frequently searched or unmatched product names to help users improve the dictionary.
7. Authentication:
   * Add user authentication and role-based access to secure the mapping system.
8. Cloud Deployment:
   * Deploy the system to a cloud service (e.g., AWS, Azure) for better availability and scalability.