MediCode Project Requirements

SRS Software Requirements Specifications
Customer
Developer
Tester
Soft. Architect

Functional Requirements

- The software must enable the (user) to enter and search for medication interactions
- API fetch
    - rxNorm
    - openFDA
- User Interface
    - Query Input
    - Return Data (Display/Rendered Data)
- Database for users to login and save information
    - Save lists of medications

Non-Functional Requirements (measurable)

- Usability
- Reliability
- Scalability
- Performance
- Portability
- Security

- The return data should clearly display and highlight interactions for each medication combination (A mentions B, B Mentions A)

Domain

- Map Arrays (Cache Data)
- Handle Interactions
- Build 3 Maps to hold query and return data
    - NormalizedCache
    - IngredientsCache
    - LabelCache

Use Cases

- User inputs list of medications
- Returned interaction data/source links
-

Project Scope

Build a web app that:
1) Accepts a user's list of medications (brand/generic)

2) Normalizes names to **RxNorm** ingredients (IN)
3) Dedupes to unique ingredients
4) Builds unique pairs and fetches label interaction text for each ingredient from openFD*
5) Flags, mentions (A label mentions B, and vice versa), and displays source snippets + links

MVP Functional Requirements

- Input
  - Enter list of drugs (comma- or newline-separated)
  - Typos tolerated via RxNorm approximate search functionality

- Normalization
  - For each input, resolve: `rxcui`, preferred display name, ingredients

- Deduplication
  - Collapse multiple brand/generics into same ingredient bucket
  - Maintain mapping: `ingredient -> {sourceDrugs[]}`

- Pairing
  - Generate unordered unique pairs of ingredients

- Label fetch
  - Per ingredient, fetch `drug_interactions[]` from openFDA (cache results)

- Matching
  - For each pair:
    - A's label text "mentions" B (substring check, min length threshold)
    - B's label text "mentions" A

- Results UI
  - Show normalized data, deduped ingredients, pairs
  - ✓ / ✗ flags with expandable snippets and **source links**
  - Status messages and a small debug pane

- Error handling
  - Continue if some entries don't match RxNorm
  - Show "no data" if no label interaction text found

Data Constructs (shapes we pass around to and from API)

We are going to cache all API data in Map Array Objects

```
Normalized Cache (Map)  = {
  query: string;
  rxcui: string | null;
```

  display: string | null;
  ingredients: string[];
  error?: "no_match";
};

- Normalized is the user input for medication names normalized, returning following properties (JSON)
- rxcui is the drug unique identifier
- display is the normalized/preferred name
- rxnorm ingredients names string array
- passes

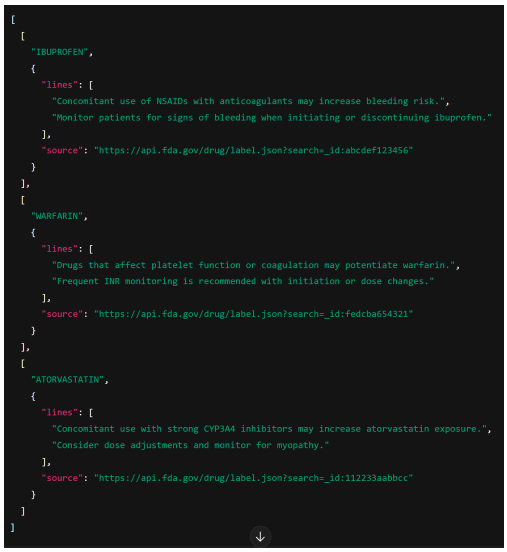
```json
{
  "query": "Advil 200 mg",
  "rxcui": "153010",
  "display": "Advil 200 MG Oral Tablet",
  "ingredients": ["IBUPROFEN"]
}
```

Ingredient Cache (Map) = {
  ingredient: string;
  key: string;
  sourceDrugs: Set<string>;   // inputs that mapped here
};

```json
[
  ["153010", ["IBUPROFEN"]],
  ["11289",  ["WARFARIN"]],
  ["83367",  ["ATORVASTATIN"]],
  ["1049630", ["ACETAMINOPHEN", "HYDROCODONE"]]
]
```

- IngredientNode returns the ingredients from the Normalized rxcui
- Maps the source drugs from the input

Label Cache (Map) = {
  lines: string[];   // openFDA label drug_interactions[]
  source: string;   // linkable query URL
};



- LabelPayload is the return from openFDA with the label drug interaction array
- Returns the source from the openFDA drug interaction

<u>Helper Functions</u>
normalizeToIngredients
getLableInteractionsByIngredient
buildIngredientIndex
buildUniquePairs
textMentions
needleSetFor
setStatus
renderDedup
renderPairRow
normalizeToIngredients

- Pings api to normalize approx input to find candidate for normalized drug name
- If no candidate, return a miss (error no match)
- Otherwise we get the rxcui (identifier)
- If candidate exists, we use the candidate to get preferred name from API
- Check inCache map for ingredients, if no ingredients, map the rxcui to ingredients from API ping
- If ingredients are there, return response and set response to map inCache

getLableInteractionsByIngredient

- Pings open FDA API for ingredient interactions
- Accepts ingredient parameter, checks if labelCache has the ingredient
- Builds array of strings for API return results of ingredient and interactions
- Returns drug interactions and source url link

buildIngredientIndex

- For deduping ingredients

- Create new map array
- If it is not in the array, adds/sets ingredient name as key and creates new set for included data
- if it is in the array gets set with key ingredient
- Returns index of Ingredients deduped

buildUniquePairs

- Accepts Ingredient Index
- Builds Ingredients array
- Double Loop through array and builds unique pairs
- Return Array with unique pairs

textMentions

- Loops through label strings to find matches for if text is mentioned in label strings
-

needleSetFor

- Builds needle set for mentioned text
- Used by textMentions to create set to compare to

setStatus

- Display function to set elements to display status (error message/done)

renderDedup

- Renders deduped ingredient lists

renderPairRow

- Displays pairs and if they mention (A mentions B, B mentions A)