Key Features for Sprint 2

Requirement 4.0: Pantry Normalization System

Feature Overview: Introduce a normalization layer for pantry ingredients so that user-entered items can be consistently matched against recipe ingredient data. This reduces mismatches caused by pluralization, casing, or naming variations.

Service Layer and Integration: Extend the pantry service to normalize ingredient names at entry time and during recipe comparisons. Normalization logic is reused across pantry search features.

Functionality:

- Pantry ingredients are stored in a normalized format.
- Ingredient names are standardized (case, pluralization, aliases).
- Normalized pantry data is used for all recipe matching logic.

Requirement 4.1: Ingredient name normalization rules

Feature Overview: Define a standardized set of rules to normalize ingredient names consistently across the system.

Structure: Create reusable normalization utilities or mappings to resolve common ingredient variants into canonical forms.

Functionality:

- Ingredient names are converted to lowercase canonical forms.
- Common aliases and plural forms resolve to the same ingredient.
- Normalization rules can be extended without refactoring services.

Requirement 4.2: Normalize pantry WITH recipe ingredients

Feature Overview: Ensure that pantry ingredient normalization is compatible with recipe ingredient data to allow accurate comparisons.

Structure: Apply normalization logic uniformly to both pantry entries and recipe ingredient lists.

Functionality:

- Pantry and recipe ingredients are compared using normalized values.
- Formatting differences no longer prevent ingredient matches.
- Existing pantry data can be reprocessed safely.

Requirement 5.0: Pantry intersection search feature

Feature Overview: Enable a search mode that filters recipes based on overlap between pantry ingredients and recipe requirements.

Service Layer and Integration: Extend the recipe search service to accept pantry data and compute ingredient intersection scores.

Functionality:

- Recipes are filtered based on pantry ingredient overlap.
- Results prioritize recipes requiring fewer missing ingredients.
- Search behavior is deterministic and repeatable.

Requirement 5.1: Create pantry-only recipe filter logic

Feature Overview: Implement core logic that calculates how well a recipe matches a user's pantry.

Structure: Add helper functions to compute ingredient intersection counts and match percentages.

Functionality:

- Intersection between pantry and recipe ingredients is computed.
- Matching thresholds can be configured.
- Logic is reusable by multiple search features.

Requirement 5.2: Add pantry intersection query mode

Feature Overview: Expose pantry-based recipe search as an explicit query mode.

Structure: Add backend query parameters and frontend controls to enable pantry-only searches.

Functionality:

- Frontend can request pantry-based search results.
- Backend routes support pantry intersection queries.
- Existing search behavior remains unaffected.

Requirement 6.0: “What Can I Make?” one-click feature

Feature Overview: Provide a one-click user experience that displays recipes based on the user's pantry contents.

UI and Flow: Add a button that triggers a pantry-based recipe search and navigates the user to results.

Functionality:

- User can initiate pantry-based search with one click.
- Results are sorted by best pantry match.
- Feature requires no additional configuration.

Requirement 6.1: Add UI button + route

Feature Overview: Add a user-facing control to access the "What Can I Make?" feature.

Structure: Introduce a new button and route that initiates the pantry-based search flow.

Functionality:

- Button is accessible from relevant UI screens.
- Clicking the button navigates to recipe results.

Requirement 6.2: Connect to pantry search service

Feature Overview: Connect the frontend interaction to the backend pantry intersection search logic.

Structure: Wire UI actions to existing search endpoints with pantry parameters.

Functionality:

- Button triggers correct API calls.
- Errors are handled using standardized error responses.

- Users can retry failed requests.

Requirement 7.0: Recipe caching layer (basic)

Feature Overview: Improve performance by caching frequently requested recipe data.

Service Layer and Integration: Introduce a lightweight in-memory cache within the backend service.

Functionality:

- Frequently requested recipe searches are cached.
- Recipe detail lookups by ID are cached.
- Cache does not compromise correctness.

Requirement 7.1: Add in-memory cache for recipe results

Feature Overview: Cache recipe search results temporarily to reduce repeated DB queries.

Structure: Store query results with a time-to-live (TTL) policy.

Functionality:

- Cache hits return results faster.
- Cache misses fall back to DB queries.

Requirement 7.2: Cache recipe details by ID

Feature Overview: Cache individual recipe detail lookups for improved performance.

Structure: Use recipe IDs as cache keys.

Functionality:

- Repeated recipe detail requests are served from cache.
- Cache invalidation does not break correctness.