Key Features for Sprint 2

Requirement 1.0: Implement User Authentication

Feature Overview: Build and integrate the user authentication system to allow users to register, log in, and recover their accounts securely.
- Schema and Service: Design and implement user database schema (e.g., Firebase Auth or custom backend solution). Define required fields and structure to support authentication and user management..

Functionality:
- Users can register, log in, and reset their passwords securely.
- Basic input validation and error messaging are in place.

Requirement 1.1: Design and set up user database schema (e.g., using Firebase Auth or a custom solution)

Feature Overview: Establish the underlying data structure to store and manage user credentials and metadata.
- Schema Design: Define attributes for authentication, profiles, and timestamps.

Functionality:
- Schema and service layer created and documented.
- Integrated with the selected authentication provider.

Requirement 1.2: Create user Registration UI screen and form

Feature Overview: Provide an interface for users to create accounts.
- UI and Flow: Registration screen with input fields for required credentials and validation states.

Functionality:
- The registration form creates valid user accounts and returns appropriate feedback.

Requirement 1.3: Create user Login UI screen and form

Feature Overview: Provide an interface for users to log in to the application.
- UI and Flow: Login screen with input fields, error messaging, and navigation after successful login.

Functionality:
- Users can log in with valid credentials and are routed to the main app.

Requirement 1.4: Implement “Forgot Password” flow and UI

Feature Overview: Allow users to recover access to their accounts through email or password reset.

- UI and Flow: “Forgot password” link on login screen, reset email flow.

Functionality:

- Reset emails sent and handled by the provider.
- User receives confirmation feedback.

Requirement 1.5: Implement input validation and error messaging for auth forms

Feature Overview: Enforce input standards and provide user feedback for errors.

- Validation Rules: Email format, password length and strength, required fields.

Functionality:

- Forms prevent submission of invalid data and display clear error messages.

Requirement 2.0: Create main UI layout

Feature Overview: Establish a consistent navigational structure and base UI template for the application.

- Structure: Navigation bar (bottom or drawer), reusable base screen, global theme, and custom header.

Functionality:

- Core app layout is consistent across screens.

Requirement 2.1: Implement core navigation structure (e.g., Bottom Nav Bar or Drawer)

Feature Overview: Provide persistent navigation across major app screens.

- Structure: Defined tabs or drawers with routing.

Functionality:

- Users can navigate between the main sections of the app.

Requirement 2.2: Create a reusable and themable base screen template

Feature Overview: Ensure all screens use a consistent structure for content.

- Structure: Header, content area, footer/controls.

Functionality:

- Base screen template applied across multiple views.

Requirement 2.3: Define and implement a global app theme (colors, typography)

Feature Overview: Unify look and feel of the application.

- Design: Color palette, typography scale, spacing system.

Functionality:

- Theme is applied globally and easy to modify in one place.

Requirement 2.4: Create a custom app header/action bar component

Feature Overview: Provide a consistent top-level UI element with navigation and action controls.

- Structure: Header with title and action buttons.

Functionality:

- Header used on all primary screens with responsive layout.

Requirement 3.0: Develop ingredient input with manual entry

Feature Overview: Allow users to enter and manage ingredients through a dynamic input system.

- Structure: Input field, suggestion system, list display, edit/remove/clear functions.

Functionality:

- Ingredients can be added, edited, and removed dynamically.

Requirement 3.1: Build a user-friendly manual input field for ingredients

Feature Overview: Provide a clean and intuitive interface for entering ingredients.

- Structure: Text field with add button and keyboard support.

Functionality:

- User can add ingredients with minimal friction.

Requirement 3.2: Implement auto-suggestions for common ingredients as user types

Feature Overview: Speed up entry by suggesting common ingredients.
- Structure: Dropdown or predictive text list.

Functionality:
- Suggestions appear as user types and can be selected.

Requirement 3.3: Implement a dynamic list to display entered ingredients

Feature Overview: Show entered ingredients in a clear list or chip view.
- Structure: List component displaying added items.

Functionality:
- List updates dynamically as items are added or removed.

Requirement 3.4: Add functionality to edit and remove items from the ingredient list

Feature Overview: Provide controls for modifying or deleting existing entries.
- Structure: Edit and delete icons/buttons on each item.

Functionality:
- User can update or delete ingredients inline.

Requirement 3.5: Add a button to clear the entire ingredient list

Feature Overview: Allow users to clear all entries at once.
- Structure: Clear all button with confirmation.

Functionality:
- Entire list is cleared after confirmation.