Key Features for Sprint 3

Requirement 4.0: Integrate with Spoonacular Recipe API

Feature Overview: Build the connection between the app and the Spoonacular API to retrieve recipe data based on user-entered ingredients and dietary preferences.
Service Layer and Integration: Create a dedicated API service responsible for building and executing HTTP requests, handling responses, and mapping results to recipe models.
Functionality:

- Recipes are fetched dynamically from Spoonacular.
- Users receive relevant recipes based on entered ingredients.
- Integration handles API/network errors gracefully.

Requirement 4.1: Set up API service layer and network client

Feature Overview: Implement the underlying network client to handle communication with the external API.
Structure: Configure endpoints, headers, and environment variables securely. Centralize fetch logic for reuse.
Functionality:

- Successful requests return structured JSON.
- Error codes and timeouts are handled appropriately.

Requirement 4.2: Parse and transform API response into app-specific recipe model objects

Feature Overview: Convert raw API responses into structured recipe model objects for display and reuse.
Data Model: Define `Recipe` class/structure with attributes such as title, image, ingredients, instructions, and nutrition data.
Functionality:

- API data is normalized into consistent objects.
- Parsed recipes populate UI components accurately.

Requirement 4.3: Implement robust error handling for API calls (e.g., no network, API limit)

Feature Overview: Ensure the app responds to connectivity issues or API limitations with helpful user feedback.
Behavior: Define user-facing error states for offline mode, timeouts, and limit exceeded cases.
Functionality:

- Users receive clear error messages.
- App remains stable and responsive during failures.

Requirement 4.4: Implement loading states (skeleton screens/spinners) during API calls

Feature Overview: Add loading indicators to improve user experience during data fetches.
UI Elements: Skeleton cards, spinners, or shimmer effects.
Functionality:

- Loading visuals appear while fetching recipes.
- Transitions smoothly to results or error states.

Requirement 5.0: Build and display recipe search results

Feature Overview: Display recipe results in an engaging, scrollable view after a successful API call.
Structure: Use reusable recipe card components within a grid or list layout.
Functionality:

- Recipes display images, titles, and details.
- Supports smooth scrolling and user navigation.

Requirement 5.1: Create a recipe card UI component to display search results

Feature Overview: Design a modular card component for recipe previews.
UI and Design: Include image, title, summary info, and Save button.
Functionality:

- Cards are reusable across screens.
- Responsive layout supports multiple device sizes.

Requirement 5.2: Build a scrollable results list/grid view

Feature Overview: Organize results into a scrollable layout for browsing.

Layout: Implement grid or vertical list with lazy loading for performance.
Functionality:

- Continuous scroll and pagination are supported.
- Users can tap a card to view full recipe details.

Requirement 5.3: Implement “No results found” state UI

Feature Overview: Provide feedback when no recipes match the search.
Design: Empty state message with optional illustration and retry option.
Functionality:

- Displays clear messaging for empty results.
- Encourages users to adjust search inputs.

Requirement 6.0: Add dietary preference features

Feature Overview: Allow users to define dietary restrictions that influence recipe searches.
Structure: Create a dedicated settings screen tied to API filters and local data.
Functionality:

- Preferences persist across sessions.
- API results align with the user’s dietary choices.

Requirement 6.1: Create a settings/preferences screen for dietary restrictions

Feature Overview: Provide UI for users to select dietary restrictions such as vegan, vegetarian, or gluten-free.
UI Design: Use toggle switches or checkbox lists for multiple selections.
Functionality:

- Preferences are saved locally or to the user profile.
- Used by the API query and filtering system.

Requirement 6.2: Integrate selected dietary filters into the Spoonacular API request

Feature Overview: Modify API requests to include the user’s dietary preferences.
Behavior: Append query parameters for chosen diet types or intolerances.
Functionality:

- Results reflect user preferences accurately.
- Dietary filters dynamically affect recipe search.

Requirement 6.3: Implement client-side filtering of manual ingredients based on dietary rules

Feature Overview: Filter ingredients entered by users to comply with selected dietary rules.
Logic: Compare ingredient entries against restricted items (e.g., meat, dairy).
Functionality:

- Warn or block incompatible ingredients.
- Users can modify entries before submitting searches.

Requirement 7.0: Develop image-based ingredient identification

Feature Overview: Enable users to capture or upload images for automatic ingredient recognition.
Integration: Combine camera/gallery access with a Computer Vision API.
Functionality:

- Ingredients detected from images.
- Recognized items are displayed for user confirmation.

Requirement 7.1: Implement image capture using device camera

Feature Overview: Let users capture photos for ingredient recognition directly within the app.
Structure: Use native device camera APIs with permission handling and preview display.
Functionality:

- Images captured securely and processed for analysis.
- Integrated into the ingredient entry workflow.

Requirement 7.2: Implement image selection from device gallery

Feature Overview: Allow users to upload existing images for ingredient analysis.
UI and Flow: Integrate gallery picker within the ingredient entry interface.
Functionality:

- Gallery access handled securely.
- Selected images sent to the recognition module.

Requirement 7.3: Integrate with a Computer Vision API (e.g., Google Vision) to identify ingredients

Feature Overview: Use an external API to identify ingredients from photos.
Implementation: Connect through SDK or REST endpoints with authentication.
Functionality:

- Returns accurate ingredient labels.
- Handles low-confidence or invalid image cases gracefully.

Requirement 7.4: Display list of identified ingredients for confirmation/editing

Feature Overview: Show detected ingredients to users for review before adding them.
UI: Editable list with checkboxes or text fields.
Functionality:

- Users can confirm, rename, or remove entries.
- Confirmed ingredients feed directly into recipe search.

Requirement 8.0: Implement recipe saving and management

Feature Overview: Allow users to save and manage their favorite recipes within the app.
Structure: Utilize a user-linked database to store saved recipes.
Functionality:

- “Save Recipe” button integrated across the UI.
- Saved recipes are viewable and removable.

Requirement 8.1: Design and implement database structure for user-specific recipe collections

Feature Overview: Create a schema to manage saved recipes tied to each user account.
Schema Design: Include recipe ID, user ID, timestamps, and metadata.
Functionality:

- Data persists across sessions.
- Supports CRUD operations.

Requirement 8.2: Add “Save Recipe” button and functionality to recipe cards

Feature Overview: Allow users to save or unsave recipes directly from search results.
UI and Flow: Use a bookmark or heart icon to toggle saved status.
Functionality:

- Immediate feedback on save state.
- Syncs with user’s recipe database.

Requirement 8.3: Create “My Saved Recipes” screen to view collection

Feature Overview: Display a user’s saved recipes in a dedicated screen.
Structure: Grid or list view similar to main results layout.
Functionality:

- Loads saved recipes from user data.
- Supports navigation to full recipe details.

Requirement 8.4: Allow users to remove recipes from their saved collection

Feature Overview: Provide the ability to delete saved recipes easily.
UI and Flow: Include delete icons, swipe gestures, or long-press actions.
Functionality:

- Recipes removed instantly with confirmation.
- Database updates reflect current saved list.