1st trial :

Random Recipe Generator

Core Challenges:

1. **API Consumption:** Connect to a recipe or food database API.
2. **Model Mapping:** The JSON response will be deep (ingredients, steps, measurements, images, nutrition). You must create a robust Dart model class (e.g., `Recipe`) with factory constructors to handle this complex mapping.
3. **Random Fetch:** Implement a "Surprise Me" button that fetches a random recipe, handling the potentially long network latency.
4. **Error States:** Implement robust checks for null or missing fields in the JSON (e.g., if an image URL is missing, display a placeholder image).
5. **Checklist Feature:** On the detail page, allow the user to tap ingredients to cross them off (local state management practice).

Recommended API Link:

- **The MealDB:** https://www.themealdb.com/api/json/v1/1/random.php

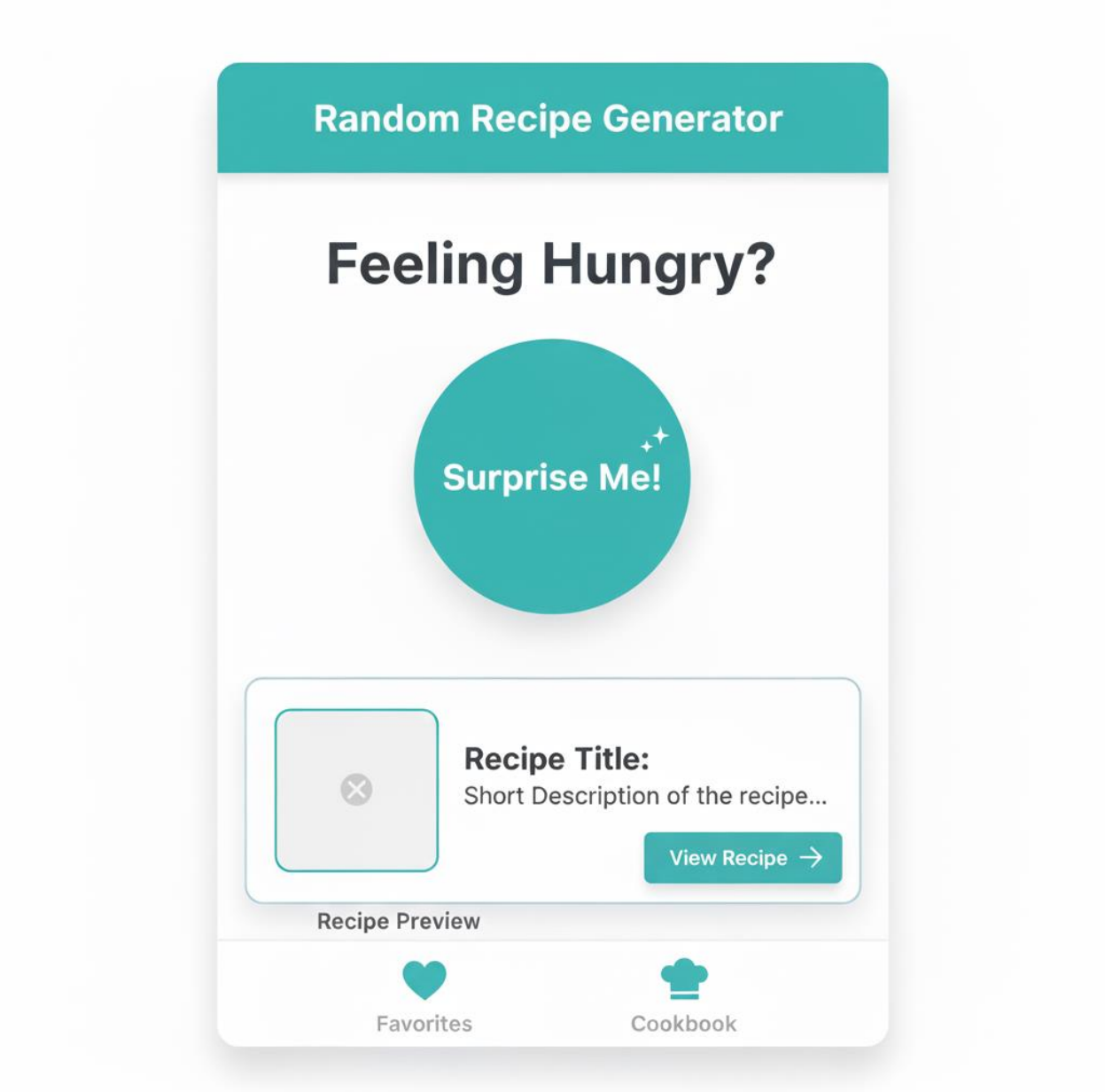
Screen 1: Home/Generator Screen Requirements

This screen manages the initial state and triggers the API call.

A. Data Flow & State Management

- **Initial State:** Show the "Surprise Me!" button and perhaps a placeholder image or empty card.
- **Loading State:** When the button is pressed, show a **CircularProgressIndicator** to indicate the network call.
- **Success State:** Once the recipe data arrives, update the UI to show the **RecipeCard**.

- **Error State:** If the API fails, display a user-friendly error message (e.g., "Could not fetch recipe, check connection.").



Screen 2: Recipe Detail Screen Requirements

This screen focuses on displaying complex, scrolling data in a clean, legible format.

A. Layout & Scrolling

- The entire screen content must be scrollable (use a `SingleChildScrollView`).
- Use a large image at the top with a `SizedBox` or specific dimensions.