Recipe Finder app

🍽️ Project: Recipe Finder App (Jetpack Compose Edition)

🔍 Core Features

- **Search Recipes**: Users can enter ingredients or keywords to search for recipes via an external API (e.g., **Spoonacular**).
- **Recipe List**: Displays a list of matching recipes with images, titles, and brief info.
- **Recipe Detail Screen**: Shows full recipe details (ingredients, steps, etc.).
- **Favorite Recipes**: Users can save favorite recipes locally for offline viewing.

🧱 Tech Stack & Components

- **Retrofit (Network Call)**:
  - Fetches recipe data from a public API like **Spoonacular** or **Edamam**.
  - Parses JSON responses into Kotlin data classes.
  - Handles query parameters for recipe search (e.g., by ingredient or cuisine).
- **Room Database (Local Storage)**:
  - Used to **store favorite recipes** with basic info and recipe IDs.
  - Supports full CRUD operations for adding and removing favorites.
  - Ensures offline access to saved recipes.
- **Jetpack Compose UI**:
  - Built with **Composable functions** for screens like:
    - **Search Input**
    - **LazyColumn-based Recipe List**
    - **Recipe Detail View**
  - Integrates images with libraries like **Coil** for loading images from URLs.
- **Navigation Compose**:
  - Manages app flow:
    → **Search Screen**
    → **Recipe List Screen**
    → **Recipe Detail Screen**
- **ViewModel + LiveData / StateFlow**:
  - Handles state and data logic.

    - Ensures clean architecture and reactive UI updates.

✅ Why It’s a Great Project

- Perfect hands-on practice for:
    - **Retrofit + API parsing**
    - **Room integration**
    - **Jetpack Compose UI patterns**
- Real-world relevance: teaches how to mix **remote and local data** effectively.
- Easy to expand:
    - Add filters (e.g. vegetarian, low-carb)
    - Include shopping lists or step-by-step cooking mode
    - User authentication to sync favorites