AI Agent Instruction:
Offline Atomic Habit Tracker (Flutter)

Architecture & Design Specifications

November 22, 2025

Abstract

This document contains the prompt instructions required to generate a fully functional, offline-first Android application using Flutter. The app focuses on habit formation (”Atomic Habits”), consistency tracking, and local data persistence.

1 Role and Objective

Role: You are a Senior Flutter Developer and UI/UX Expert specializing in Clean Architecture and local-first mobile applications.

Objective: Build a complete, production-ready Flutter Android application for tracking habits. The app must be fully offline, visually modern, and highly interactive. The code must be modular, scalable, and optimized for performance.

2 Design System & UI/UX

The application must feel modern, encouraging, and clean.

2.1 Color Palette

- **Primary Color:** #85d8ea (Light Cyan/Blue) - Used for active states, FABs, and key highlights.
- **Secondary Color:** #546a7b (Blue Grey) - Used for text, inactive icons, and headers.
- **Background:** White or very light grey to maintain contrast.
- **Feedback:** Use Green for success/completion, Red for delete/destructive actions.

2.2 Interaction & Animation

- Use **Hero Animations** when transitioning from the list view to the detail view.
- Implement a satisfying **Micro-interaction** when a user checks off a habit (e.g., a subtle scale animation, confetti, or a checkmark morph).
- UI components should have rounded corners (approx. 12-16px radius) to feel friendly.

3 Technical Architecture (Flutter Optimized)

Since the user requested Room (Native Android) and AlarmManager, you must use the **Flutter equivalents** that adhere to best practices.

3.1 Tech Stack

1. **Framework:** Flutter (Latest Stable).
2. **State Management: Riverpod** (with code generation) for robust, compile-safe state management.
3. **Local Database (The "Room" Alternative):**
   - Use **Drift** (formerly Moor). Drift is the closest Flutter equivalent to Room. It provides type-safe SQL, DAOs, and Tables, running on top of SQLite.
   - *Alternative:* If you determine Isar is faster for this specific use case, you may use Isar, but Drift is preferred for structured relational data.
4. **Notifications:**
   - Use **flutter_local_notifications** for scheduling daily reminders.
   - Use **android_alarm_manager_plus** if exact timing is critical even when the app is killed, but standard scheduled notifications are usually sufficient for habits.
5. **Architecture:** Clean Architecture (Presentation, Domain, Data layers).

4 Feature Specifications

4.1 1. Home Screen (Dashboard)

- Display a list of habits scheduled for **Today**.
- Each list item is a "Card" containing:
  - Habit Name.
  - A prominent "Check" button.
- When "Check" is clicked: Update the database, trigger animation, and visually mark as "Done".

4.2 2. Create Habit (Input Form)

- **Name:** Text input (Required).
- **Description/Motivation:** Text input (Optional).
- **Schedule Time:** Time Picker (Required). This determines when the notification fires.
- **Days:** Select specific days (Mon-Sun) or "Everyday".
- **Action:** Save to local DB and schedule the notification ID.

4.3 3. Habit Detail View

- Accessed by tapping a Habit Card on the Home Screen.
- **Stats Area:**
  - **Current Streak:** Calculate consecutive days completed.
  - **Total Completions:** Total count.
- **Calendar View:** Display a monthly calendar (using `table_calendar` package or similar). Highlight days where the habit was completed with the Primary Color.

5 Data Structure Schema (Drift/SQL)

Table: Habits

- `id` (Int, AutoIncrement)
- `title` (Text)
- `description` (Text, Nullable)
- `reminderTime` (DateTime/String)
- `targetDays` (List/String - e.g., "1,2,3,4,5,6,7")

Table: HabitEntries (The logs)

- `id` (Int, AutoIncrement)
- `habitId` (Int, Foreign Key)
- `date` (DateTime - stripped of time)
- `isCompleted` (Bool)

6 Code Generation Instructions

When generating the code:

1. **Folder Structure:** Organize by feature (e.g., `lib/features/habit/data`, `lib/features/habit/presenta`
2. **Comments:** Comment complex logic, especially the Streak Calculation algorithm.
3. **Assets:** Use `flutter_launcher_icons` for app icon setup logic. Use `google_fonts` for typography.
4. **Responsiveness:** Ensure the layout works on different Android screen sizes.