Weight Loss Coach – SwiftUI App Plan

A private-by-default iPhone app to guide daily weight-loss habits and let you talk to an on-device coach.

---

1) Product Goals

- Make the *next healthy choice* easy with tiny, repeatable actions.
- Capture voice/text reflections securely on device.
- Offer a conversational coach using a local LLM (no cloud required).
- Keep the UI calm, fast, and low-friction (≤30 seconds to log a day).

---

2) MVP Feature Set

1) **Night Prep / Morning Focus Worksheet** - Pre-fill checklist + text fields exactly from your template. - Single screen for Night Prep; single screen for Morning Focus. - Optional End-of-Day check-in.

2) **Daily Entries & Streaks** - One entry per date (editable throughout the day). - Streak count based on any logged action.

3) **Quick Capture** - Large button: “I’m craving / I’m stressed” → opens a 20-second voice note + prompt to choose a swap.

4) **Coach Chat (Local LLM)** - Simple chat UI. - On-device model with system prompt tuned to your goals. - Can reference today’s entry (your why, chosen swap) when replying.

5) **Reminders & Widgets** - Night Prep reminder (9:00 PM default), Morning Focus (8:00 AM default). - Lock Screen/Home Screen widgets: “Log Night Prep”, “Start Morning Focus”, “Talk to Coach”.

6) **Privacy** - Everything stored locally (SwiftData). Optional iCloud sync (off by default). - On-device speech recognition where supported.

---

3) Architecture Overview

- **UI**: SwiftUI
- **Persistence**: SwiftData (iOS 17+) or Core Data fallback if needed.
- **Speech**: `SFSpeechRecognizer` with `supportsOnDeviceRecognition == true` when available; otherwise standard path.
- **LLM** (choose 1 to start, keep interface swappable):
- **Option A — llama.cpp** (Metal backend). Ship a small quantized model (e.g., 1.5–3B) as an in-app resource or downloadable asset.

- **Option B — MLC-LLM (TVM)**: iOS-friendly runtime with Metal acceleration and model packs.
- **Option C — System Intelligence (when available)**: If future iOS exposes sanctioned on-device models via API, you can add an adapter.

Create a `LocalLLMService` protocol so you can swap implementations without touching UI.

---

4) Data Model (SwiftData)

```
import SwiftData
import Foundation

@Model
final class DailyEntry {
    @Attribute(.unique) var id: UUID = UUID()
    var date: Date = .now

    // Night Prep
    var stickyNotes: Bool = false
    var preppedProduce: Bool = false
    var waterReady: Bool = false
    var breakfastPrepped: Bool = false
    var nightOther: String = ""

    // Morning Focus
    var myWhy: String = ""
    var challenge: Challenge = .none
    var challengeOther: String = ""
    var chosenSwap: String = ""
    var commitFrom: String = ""   // instead of ___
    var commitTo: String = ""     // Today I will ___ instead of ___

    // End of Day
    var followedSwap: Bool? = nil
    var feelAboutIt: String = ""
    var whatGotInTheWay: String = ""

    // Voice notes (file URLs in app sandbox)
    var voiceNotes: [URL] = []

    init() {}
}

enum Challenge: String, Codable, CaseIterable, Identifiable {
    case none, skippingMeals, lateNightSnacking, sugaryDrinks, onTheGo,
emotionalEating, other
```

```
    var id: String { rawValue }
}
```

If you prefer Core Data, mirror the same fields in an `NSManagedObject` subclass.

---

5) UI Map

- **Tab 1: Today**
- Night Prep (toggle checklist + free text)
- Morning Focus (why, choose challenge, choose swap, commit)
- End-of-Day Check-In
- Big button: “Quick Capture” (voice/text)
- **Tab 2: Coach**
- Chat interface; messages persisted locally.
- **Tab 3: History**
- Calendar list of entries; tap to view/edit.
- **Tab 4: Settings**
- Reminders, iCloud sync toggle, export/import, model selection/size.

---

6) SwiftUI Screens (Skeletons)

TodayView

```
struct TodayView: View {
    @Environment(\.modelContext) private var context
    @Query(sort: \DailyEntry.date, order: .reverse) private var entries:
[DailyEntry]

    @State private var entry: DailyEntry = DailyEntry()

    var body: some View {
        ScrollView {
            NightPrepSection(entry: $entry)
            Divider().padding(.vertical)
            MorningFocusSection(entry: $entry)
            Divider().padding(.vertical)
            EndOfDaySection(entry: $entry)

            Button(action: quickCapture) {
                Label("Quick Capture", systemImage: "mic.circle.fill")
                    .font(.title2)
                    .padding()
            }
```

```
            .buttonStyle(.borderedProminent)
            .padding(.top)
        }
        .navigationTitle("Today")
        .onAppear { loadOrCreateToday() }
        .toolbar { SaveButton(entry: entry) }
    }

    private func loadOrCreateToday() {
        let startOfDay = Calendar.current.startOfDay(for: Date())
        if let existing = entries.first(where: {
Calendar.current.isDate($0.date, inSameDayAs: startOfDay) }) {
            entry = existing
        } else {
            entry = DailyEntry()
            entry.date = startOfDay
            context.insert(entry)
            try? context.save()
        }
    }

    private func quickCapture() {
        // present a sheet with voice/text capture
    }
}
```

Night Prep Section

```
struct NightPrepSection: View {
    @Binding var entry: DailyEntry

    var body: some View {
        VStack(alignment: .leading, spacing: 12) {
            Text("Night Prep (5 minutes)")
                .font(.title3).bold()
            Toggle("Put sticky notes where I usually grab the less-healthy
choice", isOn: $entry.stickyNotes)
            Toggle("Wash/cut veggies or fruit and place them at eye level",
isOn: $entry.preppedProduce)
            Toggle("Put water bottle in fridge or by my bed", isOn:
$entry.waterReady)
            Toggle("Prep quick breakfast/snack", isOn: $entry.breakfastPrepped)
            TextField("Other…", text: $entry.nightOther)
                .textFieldStyle(.roundedBorder)
        }
        .padding()
```

```
        .background(RoundedRectangle(cornerRadius:
16).fill(Color(.secondarySystemBackground)))
    }
}
```

Morning Focus Section

```
struct MorningFocusSection: View {
    @Binding var entry: DailyEntry
    var body: some View {
        VStack(alignment: .leading, spacing: 12) {
            Text("Morning Focus (10 minutes)").font(.title3).bold()

            // Step 1 – My Why
            Text("Step 1 – My Why (2 minutes)").bold()
            TextEditor(text: $entry.myWhy).frame(minHeight: 80)
                .overlay(RoundedRectangle(cornerRadius: 8).stroke(.quaternary))

            // Step 2 – Identify a Challenge
            Text("Step 2 – Identify a Challenge (3 minutes)").bold()
            Picker("Challenge", selection: $entry.challenge) {
                Text("Select…").tag(Challenge.none)
                Text("Skipping meals").tag(Challenge.skippingMeals)
                Text("Late-night snacking").tag(Challenge.lateNightSnacking)
                Text("Sugary drinks").tag(Challenge.sugaryDrinks)
                Text("Eating on the go / fast food").tag(Challenge.onTheGo)
                Text("Emotional eating").tag(Challenge.emotionalEating)
                Text("Other").tag(Challenge.other)
            }.pickerStyle(.menu)

            if entry.challenge == .other {
                TextField("Describe the challenge…", text:
$entry.challengeOther)
                    .textFieldStyle(.roundedBorder)
            }

            // Step 3 – Choose My Swap
            Text("Step 3 – Choose My Swap (3 minutes)").bold()
            TextField("What healthier choice will I do instead?", text:
$entry.chosenSwap)
                .textFieldStyle(.roundedBorder)

            // Step 4 – Commit
            Text("Step 4 – Commit (2 minutes)").bold()
            Text("Today I will … instead of
…").font(.subheadline).foregroundStyle(.secondary)
```

```
            HStack {
                TextField("do this…", text: $entry.commitTo)
                Text("instead of")
                TextField("not this…", text: $entry.commitFrom)
            }
            .textFieldStyle(.roundedBorder)
        }
        .padding()
        .background(RoundedRectangle(cornerRadius:
16).fill(Color(.secondarySystemBackground)))
    }
}
```

End-of-Day Section

```
struct EndOfDaySection: View {
    @Binding var entry: DailyEntry
    var body: some View {
        VStack(alignment: .leading, spacing: 12) {
            Text("End-of-Day Check-In (Optional)").font(.title3).bold()
            Toggle("I followed my swap", isOn: Binding(
                get: { entry.followedSwap ?? false },
                set: { entry.followedSwap = $0 }
            ))
            TextField("If yes, how do I feel about it?", text:
$entry.feelAboutIt)
                .textFieldStyle(.roundedBorder)
            TextField("If no, what got in the way?", text:
$entry.whatGotInTheWay)
                .textFieldStyle(.roundedBorder)
        }
        .padding()
        .background(RoundedRectangle(cornerRadius:
16).fill(Color(.secondarySystemBackground)))
    }
}
```

---

7) Coach Chat (Local LLM) – Interfaces

Abstraction

```
protocol LocalLLMService {
    func generateReply(to messages: [LLMMessage], context: CoachContext) async
```

```
throws -> String
}

struct LLMMessage: Identifiable, Codable {
    enum Role: String, Codable { case user, assistant, system }
    var id: UUID = UUID()
    var role: Role
    var content: String
    var timestamp: Date = .now
}

struct CoachContext: Codable {
    var todayWhy: String
    var todaySwap: String
    var commitTo: String
    var commitFrom: String
}
```

System Prompt (first message)

```
You are a compassionate, pragmatic weight-loss coach. Focus on tiny, doable
actions and pattern awareness.
Use the user's daily context (why, chosen swap, commitment). Avoid shame. Offer
one concrete next step.
Keep replies under 120 words unless asked.
```

Mock Implementation (for development)

```
final class MockLLM: LocalLLMService {
    func generateReply(to messages: [LLMMessage], context: CoachContext) async
throws -> String {
        let last = messages.last?.content ?? ""
        return "I hear you. Given your commitment to \
(context.commitTo) instead of \(context.commitFrom), try one tiny step: drink
water and set a 5-minute timer before deciding. What feels doable right now
about: \(last.prefix(80))…?"
    }
}
```

Hooking Up a Real On-Device Model

- **llama.cpp route**
- Add a dependency (static library or Swift Package) that exposes a simple `predict(prompt: String)` Metal-accelerated call.

- Bundle a small quantized model (e.g., 2–4 GB can be too large; consider 1–2 GB or offer in-app download after consent).
- Stream tokens and surface partial text in the UI.

- **MLC-LLM route**

- Integrate its iOS runtime and select a compact model variant.
- Use Metal for GPU acceleration on device; control context length/token limits to protect battery.

App Store note: if you download models post-install, present clear consent and allow deletion in Settings.

---

8) Speech – Private Voice Notes

- Use `AVAudioSession` + `AVAudioRecorder` for raw voice notes stored locally.
- For transcription, use `SFSpeechRecognizer`:
- Check `supportsOnDeviceRecognition` and prefer on-device if available/locale supported.
- Provide a toggle in Settings: "Transcribe voice notes automatically".

```
func requestSpeechAuth() async throws {
    let status = await SFSpeechRecognizer.authorizationStatus()
    if status != .authorized { _ = await
SFSpeechRecognizer.requestAuthorization() }
}
```

---

9) Notifications & Widgets

- **Reminders**: `UNUserNotificationCenter` with categories: Night Prep, Morning Focus, End-of-Day.
- **Widgets**: Activity summarizer + quick actions. Deep link to Today/Coach via `widgetURL`.

---

10) Settings

- iCloud Sync (SwiftData + CloudKit) toggle.
- Model Management: choose installed model; show size; delete model.
- Export: JSON export of entries; Import for restore.

---

12) Gamification & Motivation

Why: Positive reinforcement and small rewards increase user retention and habit adherence (streaks/rewards often boost engagement materially). Keep it optional and kind.

Features

- **Mini-Milestones**: Badges/celebrations for 3-day, 7-day, 14-day, 30-day streaks; also first week of Morning Focus completed.
- **Lightweight Confetti**: Small celebration overlay when a milestone is hit. (Keep it subtle; user can disable in Settings.)
- **Progress Visualization**: Weekly completion ring + streak heatmap in History.
- **Positive Nudges in Coach**: The coach can read streak/weekly progress and add one encouraging sentence.

Data Model

```
@Model
final class Achievement {
    @Attribute(.unique) var id: UUID = UUID()
    var name: String
    var dateEarned: Date
    var type: String // e.g., "streak", "consistency", "first_week"
    var details: String

    init(name: String, dateEarned: Date = .now, type: String, details: String =
"") {
        self.name = name
        self.dateEarned = dateEarned
        self.type = type
        self.details = details
    }
}
```

Streak Heatmap (History)

```
struct StreakHeatmap: View {
    let entryDates: Set<Date> // normalized to startOfDay
    let weeksToShow: Int = 12

    private var weeks: [[Date]] {
        let cal = Calendar.current
        let today = cal.startOfDay(for: Date())
        // Build an array of weeks (Sun..Sat or Mon..Sun per locale)
        var result: [[Date]] = []
```

```
        let weekday = cal.component(.weekday, from: today)
        let startOfThisWeek = cal.date(byAdding: .day, value: -(weekday-1), to:
today) ?? today
        for w in 0..<weeksToShow {
            let start = cal.date(byAdding: .day, value: -(w*7), to:
startOfThisWeek)!
            let days = (0..<7).compactMap { cal.date(byAdding: .day, value: $0,
to: start) }
            result.append(days)
        }
        return result.reversed()
    }

    var body: some View {
        VStack(alignment: .leading) {
            Text("Streaks")
                .font(.headline)
            Grid(alignment: .leading, horizontalSpacing: 4, verticalSpacing: 4)
{
                ForEach(weeks, id: \.self) { week in
                    GridRow {
                        ForEach(week, id: \.self) { day in
                            let hit =
entryDates.contains(Calendar.current.startOfDay(for: day))
                            Rectangle()
                                .fill(hit ? Color.teal :
Color.gray.opacity(0.2))
                                .frame(width: 12, height: 12)
                                .cornerRadius(3)
                                .accessibilityLabel(Text(hit ? "Completed" :
"Not completed"))
                        }
                    }
                }
            }
        }
    }
}
```

Weekly Completion Ring (Today/History)

- Compute fraction of the last 7 days with any logged action.
- Display with `Circle().trim(from: 0, to: progress)` and a small label.

Coach Nudges Integration

Add streak context to the `CoachContext` and inject one encouraging sentence when thresholds are met.

```
struct CoachContext: Codable {
    var todayWhy: String
    var todaySwap: String
    var commitTo: String
    var commitFrom: String
    var currentStreak: Int
    var daysThisWeek: Int
}

extension LocalLLMService {
    func nudge(for context: CoachContext) -> String? {
        if context.currentStreak == 7 { return "Seven days in a row—amazing
consistency!" }
        if context.daysThisWeek >= 5 { return "You've showed up on 5 days this
week—your future self feels this." }
        return nil
    }
}
```

Settings

- Toggle: **Show celebrations** (on by default).
- Toggle: **Show streak widgets**.
- Button: **Reset achievements** (with confirm).

---

11) Phased Build Plan

Phase 0 (Day 1–2) - Project setup, SwiftData model, TodayView skeleton, local saves.

Phase 1 - Night Prep + Morning Focus + End-of-Day fully working. - Notifications for Night/Morning. - History list & detail.

Phase 2 - Quick Capture voice notes; optional transcription. - Widgets.

Phase 3 - Coach Chat using `MockLLM` → swap to real on-device model via llama.cpp or MLC-LLM. - Settings: model management, iCloud sync.

---

12) Design Notes

- Typography: large friendly headings; buttons that read like "actions" (verbs).
- Color: soothing neutrals with one accent (e.g., teal) for primary actions.
- Empty states with a single encouraging sentence.

13) Stretch Ideas (Later)

- Tie-ins with Apple Health: weight, steps (read-only with consent) → trend nudges.
- On-device intent detection (classify “craving”, “stress”, “boredom”).
- PDF export of weekly reflections.
- “Emergency plan” button that surfaces your top 3 swaps and a 90-second calming audio.

---

14) Next Steps (Concrete)

1) Create Xcode project (iOS 17+), add SwiftData model from §4. 2) Build TodayView from §6 and verify save/load for current day. 3) Add two local notifications for Night Prep and Morning Focus. 4) Implement `MockLLM` and chat UI scaffold; confirm interface. 5) Choose model path (llama.cpp or MLC-LLM) and budget app size. 6) Add voice capture with file storage; add optional transcription.

> When you’re ready, we can drop in a real LLM adapter—your UI won’t have to change.