Title: Daily Tracking App - Folder and File Structure Plan

Summary: This document outlines the complete project folder and file structure for the Daily Tracking App built in React Native. The structure is optimized for maintainability, modularity, and support for local SQLite storage, OpenAI integration, and clean UI organization based on Style Guide Set 1.

Goal

Ensure scalable, maintainable code organization from day one by separating presentation, logic, state, and service layers.

Root-Level Structure

/App.tsx ← Main app entry (navigation + context setup)
/package.json
/babel.config.js
/tsconfig.json

/components

Reusable, stateless UI components.

/Button.tsx
/MoodSlider.tsx
/TrackableCard.tsx
/SubPageInputGroup.tsx
/InsightCard.tsx
/HeaderBar.tsx

> These components take props and contain no business logic.

/screens

High-level pages tied to routes.

/Onboarding/
/Dashboard/
/SearchTrackables/
/SubPageForm/
/ManageTrackables/
/HistoryOverview/
/HistoryDetail/
/Insights/
/Settings/

Each screen folder can include:

/index.tsx ← Screen component
/styles.ts ← Local styles
/hooks.ts (optional) ← Local screen logic hooks

/services

Backend integrations, APIs, and database logic.

/db/
  schema.ts ← SQLite table definitions
  setup.ts ← DB init and connection
  queries.ts ← Select/Insert/Update helpers

/ai/
  openaiClient.ts ← OpenAI API config + requests
  promptBuilder.ts ← Structured prompt logic

/firebase/ (v2)
  firebaseClient.ts

/contexts

React Context for global state management.

/TrackablesContext.tsx
/MoodContext.tsx
/SettingsContext.tsx

> For shared state across screens (e.g., daily values, UI preferences).

/navigation

App navigation flow using React Navigation.

/index.tsx
/AppNavigator.tsx
/AuthNavigator.tsx (future use)

/utils

Helper functions and constants.

/formatDate.ts
/parseMoodScore.ts
/constants.ts
/colors.ts
/theme.ts

/assets

Static resources.

/icons/

/fonts/
/illustrations/

/styles (optional)

Centralized shared styles.

/global.ts
/buttons.ts
/forms.ts

> Used if you don’t want to inline styles or use styled-components.

Optional Future Folders

Folder	Use Case
/hooks	Custom reusable logic (e.g. useTrackables)
/middleware	Offline queues, sync layers (e.g. pending AI requests)
/tests	Jest unit/integration test coverage

Summary of Principles

Principle	Outcome
Separation of Concerns	Screens = page flow; Components = UI building blocks
Modularity	Features like DB, AI, and state are compartmentalized
Reusability	Shared logic lives in utils, components, or hooks

Offline-First	DB logic and contexts allow full offline functionality