1. CORE STACK (Already Defined)

- Framework: React Native + TypeScript

- Navigation: React Navigation (Stack + Bottom Tabs)

- State Management: Zustand

- Data fetching & caching: @tanstack/react-query

- Camera & Barcode/QR Scanning: react-native-vision-camera + vision-camera-code-scanner

- Font: Space Grotesk (default app-wide)

- Theme Color: Green (#4CAF50)

- Local Database: SQLite or Realm (offline-first, food log, kcal tracking)

- Backend: Express.js + MongoDB (sync data when online)

- Folder Structure (Clean Architecture, already defined)

2. MANDATORY FEATURES (Core Product)

🔹 Authentication & User Management (react-native-app-auth)

- Workflow

- Sign Up → After creating an account, the user is redirected to a Welcome / Onboarding flow with about 3–5 steps introducing the app’s main features. Once completed, the user proceeds to the Home screen.

- Sign In → Direct access to the Home screen after logging in.

- OAuth (Google / Apple / Facebook)

- Check if the user exists in the database.

- If exists → Sign In flow (Home).

- If not found → Sign Up flow (Onboarding → Home).

- Features

- Login / Signup with Email + Password

- OAuth login (Google / Apple / Facebook)

- User profile picture upload

- Secure session handling with JWT or Secure Storage

🔹 Profile Settings

- Basic profile info (name, age, height, weight, gender)

- Nutrition goals (target kcal, macros)

- Dark mode toggle

- Language localization (multi-language support)

🔹 Food Log & Tracking

- TDEE & BMI & BMR ...etc

- Search database of foods (API like Open Food Facts)

- Edit/Delete scanned items

- Add custom food manually (name, kcal, macros)

- Daily/weekly summary of kcal intake

- Export food log (CSV, PDF, Share)

🔹 Core App Flow

- Bottom tabs: Home, Scan, Log, Profile

- Vision Camera integration for scanning

- Zustand store for runtime state (user info, session)

- React Query Client for API management

- Database Schema: FoodItem(id, name, kcal, barcode, scannedAt)

- Workflow: scan → save to local DB → update food log

- Global styling: Space Grotesk font + green theme

- TypeScript path alias configuration

- Scripts in package.json (dev, build, debug)

- Basic backend folder structure (Express + MongoDB)

3. OPTIONAL FEATURES (Enhancements)

🔹 Nutrition Analytics

- Macro breakdown (protein, carbs, fats)

- Visual charts (pie, bar, trends over time)

- Personalized calorie goal tracking

- Water intake tracker

🔹 Camera & Scanning Enhancements

- Auto-focus & continuous scan mode

- Detect duplicate scans & warn user

- Support for custom QR codes (meal plans, recipes)

- Offline recognition (cached food DB)

🔹 Notifications & Reminders

- Daily reminder to log meals

- Smart notifications based on missing logs

- Push notifications

🔹 Social & Community Features

- Share food log with friends

- Community challenges (30-day calorie tracking, etc.)

- Leaderboards or streak tracking

🔹 Backend Enhancements

- Nutrition API integration (fetch nutrition facts)

- Data analytics API (recommendations, trends)

- Admin dashboard (manage users, logs, analytics)

🔹 AI Chat & Analysis

- In-app AI assistant for nutrition Q&A

- Example interactions:

• "Hôm nay ăn vậy có vượt kcal không?"

• "Gợi ý bữa tối ít carb giúp tôi"

- Summarize food log → gửi AI API → trả lời chat

- Optional: offline model hoặc rule-based chatbot

4. DEVELOPMENT WORKFLOW

- Setup dependencies

- Link fonts (Space Grotesk)

- Configure camera permissions (iOS + Android)

- Implement navigation

- Setup Zustand + React Query

- Create database schema & connection (SQLite/Realm)

- Implement authentication flow

- Build scanning & log flow (MVP)

- Add Profile Settings (mandatory)

- Expand with analytics, notifications, AI Chat (optional)