MomCare Expo

MomCare is an Expo-based prenatal companion leveraging Retrieval-Augmented Generation, multimodal AI, and Supabase Edge Functions to deliver evidence-backed guidance and culturally-resonant support to busy urban mothers. Key features include streaming text and real-time voice conversations with inline citations, GPT-4o-powered meal and posture analysis, interactive tracking for symptoms, fetal kicks, nutrition, and health alerts with comprehensive haptic feedback, bilingual support (Hindi/English), instant login state caching for seamless app reopens, immersive full-screen image backgrounds on onboarding and login screens, and a curated resource discovery hub—all secured through Row Level Security and unified Edge Function architecture. Innovation lies in combining OpenAI's Realtime API for conversational voice, vector-based document memory for context-aware responses, and on-device tool resolution to personalize clinical insight while maintaining privacy. This approach ensures mothers receive proactive prenatal care that feels warm, accessible, and intelligent, impacting maternal health outcomes by reducing anxiety and improving adherence to clinical guidance. Scalability is built into the serverless Supabase infrastructure, enabling seamless integration of future enhancements like push notifications, clinician collaboration dashboards, adaptive trimester-based care plans, offline-first journaling, moderated peer support communities, expanded multilingual capabilities, and automated testing pipelines—all while keeping the mobile client lean and the backend secure, compliant, and ready for analytics-driven insights.

System Requirements

- Node.js 18.18 LTS or later (`node --version`)
- npm 10.3 or later (`npm --version`)
- Expo CLI 6.3 or later (`npm install -g expo-cli` and `npx expo --version`)
- Java Development Kit 17 with Android SDK 34 for Android builds; Xcode 15+ for iOS simulation (macOS only)
- Physical device or simulator/emulator with microphone access for voice features
- 64-bit Windows, macOS, or Linux machine with at least 8 GB RAM and 10 GB free disk space

Getting Started

1. Confirm prerequisites are installed and at the versions above. Install the Expo Go or Dev Client app on any companion devices you plan to test with.
2. Install dependencies from the project root:

```
npm install
```

3. Configure environment variables (via `.env`, Expo secrets, or CI variables):

- `EXPO_PUBLIC_SUPABASE_URL`
- `EXPO_PUBLIC_SUPABASE_KEY`
- `SUPABASE_SERVICE_ROLE_KEY`
- `OPENAI_API_KEY`
- Optional: `DATABASE_URL` for local Supabase CLI tasks.

4. Start the Expo development server:

```
npm start
```

Press `i` for iOS simulator, `a` for Android emulator, or scan the QR code with the Expo client.

5. Deploy or update Edge Functions with the Supabase CLI whenever backend code changes.

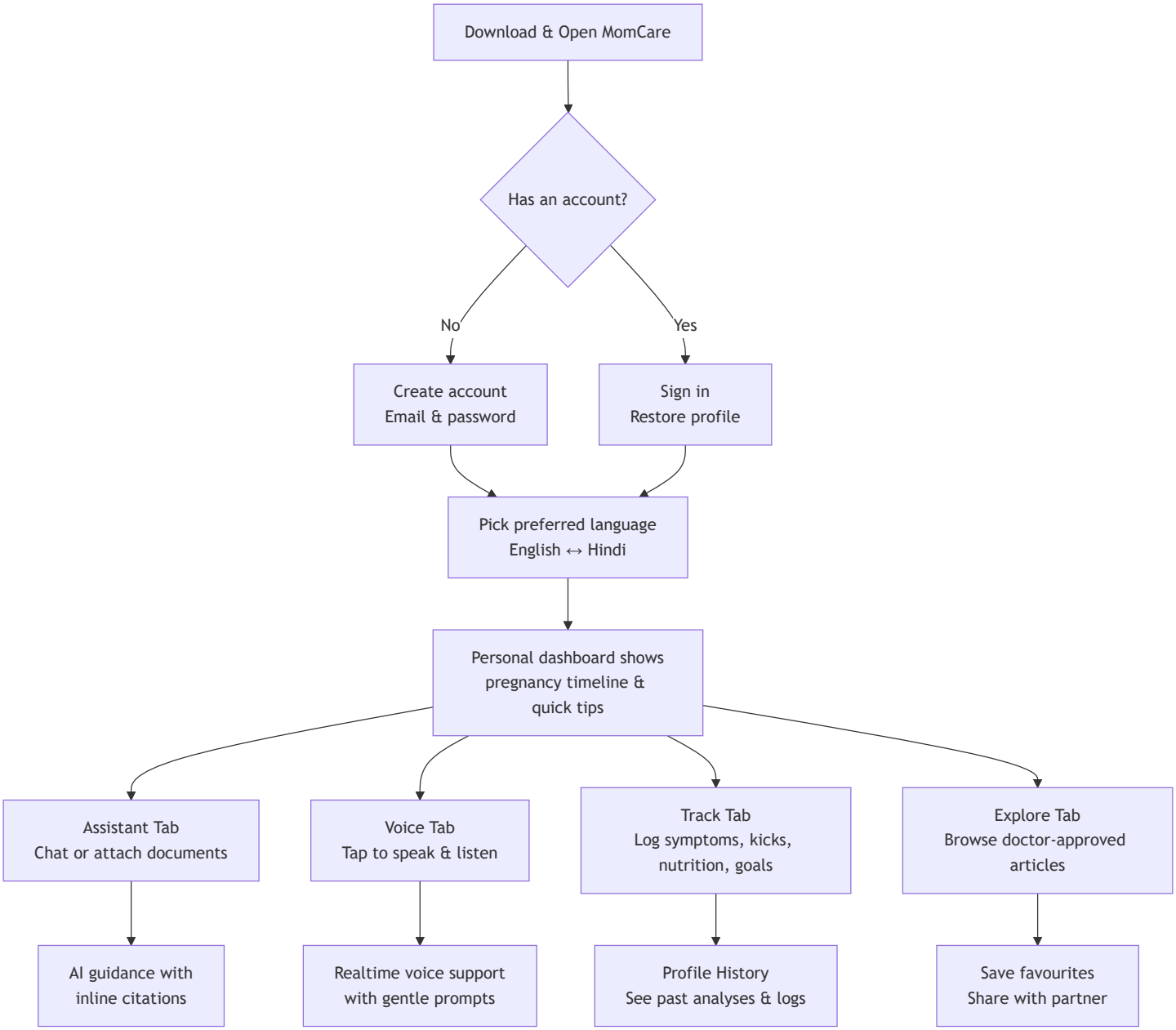
Working Capabilities

- Authentication: email/password signup, password reset links, instant session restoration via AsyncStorage caching (bypasses "getting started" screen for returning users), and automatic profile bootstrap with language preference loading.
- Assistant (text): streaming chat in Hindi and English, inline citations, document-aware memory, attachment ingestion via Edge Functions, and haptic feedback on all interactions.
- Assistant (voice): realtime WebRTC conversations backed by the OpenAI Realtime API, on-device tool resolution, dynamic prompts informed by user logs, and tactile feedback during streaming.
- Tracking: interactive screens for symptoms, fetal kicks, nutrition logs, personal goals, and priority-based health alerts with comprehensive haptic feedback on buttons, toggles, date pickers, and keyboard interactions—all persisted through Supabase Edge Functions.
- Image analysis: GPT-4o-powered meal and posture evaluation with automatic nutrition entries and history surfaced in the profile screen.
- Discovery: Explore tab delivering curated resources with search, filters, bookmarking, sharing flows, and haptic feedback on tab switches and interactions.
- Visual design: enhanced Material Design-inspired theme with darker, more saturated colors for better contrast; full-screen image backgrounds on onboarding and login screens with optimized overlays for text visibility.
- Platform plumbing: unified `lib/supabase-api.ts` client mediating every network request, `lib/cache-manager.ts` for instant session restoration, and background job execution via

Supabase Edge Functions.

User Journey At A Glance

The flow below highlights what a typical mom experiences inside the app, from sign-in to ongoing support.



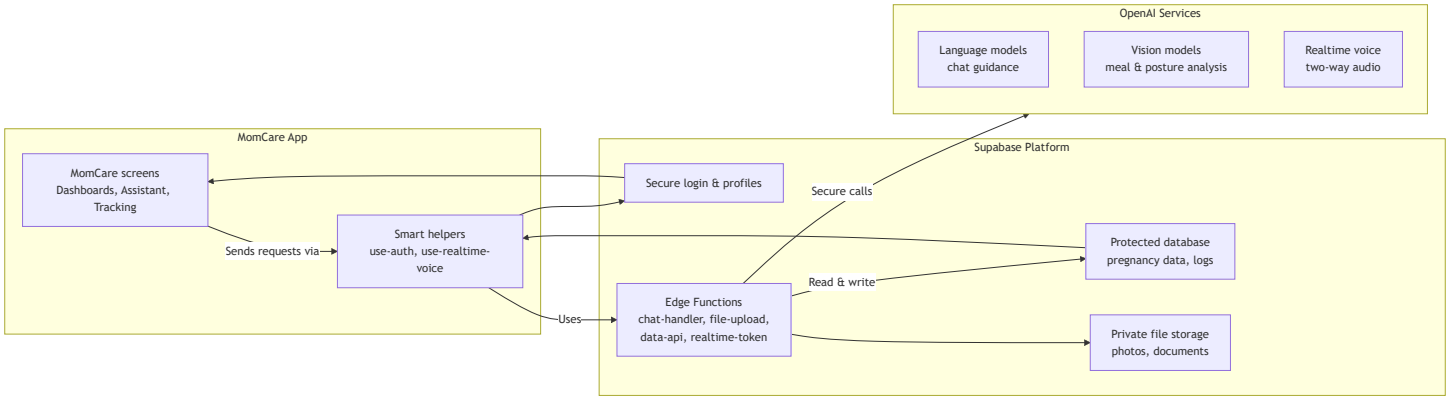
What the user sees

- **Clear guidance**: Each screen focuses on one job—assist, track, explore, or review progress.
- **Always available**: Chat and voice assistants respond in either Hindi or English and explain the source of information.

- **Celebrates progress**: Profile history surfaces meal analyses, posture insights, and pregnancy milestones to keep motivation high.

System Architecture Snapshot

This diagram explains how the mobile app, Supabase services, and OpenAI work together without requiring coding knowledge.



Why this setup works

- **Privacy first**: Sensitive OpenAI keys and health data stay inside Supabase; the phone never stores them.
- **One gateway**: Edge Functions act as the single gatekeeper, so every feature follows the same rules and security checks.
- **Performance & scale**: Supabase handles authentication, storage, and database logic, while OpenAI delivers chat, voice, and vision intelligence.

Architecture Overview

- **Expo Client**: Expo Router, TypeScript, theming in `MotherhoodTheme` with enhanced Material Design colors, AsyncStorage session handling with instant cache restoration, expo-haptics integration across 50+ interaction points.
- **Auth & Data**: Supabase Postgres with pgvector plus strict Row Level Security across user tables.
- **Edge Functions**:
  - `chat-handler` — text assistant with memory and inline citations.
  - `chat-attachments` — document ingestion, summarisation, and embeddings.
  - `data-api` — CRUD gateway for tracking modules and profile data.
  - `file-upload` — GPT-4o vision pipeline with nutrition logging.
  - `realtime-token` — OpenAI Realtime session broker for WebRTC voice.

    - `background-job-worker` — executes queued jobs and alerts.
- **AI Providers**: OpenAI GPT-4o/GPT-4o-mini, text-embedding-3-small, Whisper transcription, Realtime streaming.
- **Hooks**: `use-auth`, `use-chat-retry`, `use-image-analysis`, `use-realtime-voice`.
- **Utilities**: `lib/cache-manager.ts` for session persistence, `constants/storage.ts` for AsyncStorage keys, `constants/theme.ts` for unified color palette.

Directory Highlights

```
app/                    Expo Router screens (tabs, auth, onboarding with full-screen images)
components/             Reusable UI and assistant modules with haptic feedback
hooks/                  Client-side state and data hooks
lib/supabase-api.ts     Unified Edge Function client
lib/cache-manager.ts    Session caching for instant app loads
constants/theme.ts      Material Design color palette and spacing
constants/storage.ts    AsyncStorage key definitions
supabase/functions/     Edge Function source
docs/                   Feature status and implementation notes
```

Useful Scripts

- `npm start` – Expo development server.
- `npm run android` / `npm run ios` – Build and launch on a device or simulator.
- `npm run lint` – Lint the project via ESLint.
- `npm run reset-project` – Clear caches for stubborn Expo builds.

Development Notes

- Voice features require a device (or simulator with a microphone) and microphone permissions.
- Haptic feedback requires physical device testing; simulators/emulators may not provide tactile feedback.
- Login state is cached locally via AsyncStorage; clear app data to test first-time onboarding flow.
- Attachments are stored in the `conversation-files` bucket; align storage policies between dev and prod.
- Image analysis expects files under roughly 5 MB and persists results to `image_analysis_results` (plus `nutrition_logs` for meals).

- The assistant relies on `conversation_documents` and `document_embeddings`; run migrations before exercising document uploads.
- Onboarding and login screens use images from `assets/onboarding_images/`; ensure these assets exist before building.

Roadmap

- Analytics dashboards and observability for Edge Function usage.
- Push notifications for reminders and clinician escalations.
- Broader multilingual support beyond Hindi/English (additional TTS voices).
- Automated integration tests for Edge Functions and tracking workflows.

For status snapshots and deeper implementation details, see `docs/feature_status_docs/FEATURE_STATUS.md` and `docs/feature_status_docs/QUICK_REFERENCE.md`.

Future Enhancements

- Deeper clinician collaboration features with secure message review and escalation loops.
- Adaptive care plans that adjust prompts and reminders based on trimester milestones.
- Offline-first journaling so moms can capture insights without a signal and sync later.
- Peer support moments powered by moderated community spotlights and shared victories.