PetitPal MVP — Developer Implementation Guide

Audience: mobile devs (Flutter/Dart). Goal: get productive fast with zero ambiguity.

0. Purpose & Scope

PetitPal is a voice-first assistant for seniors. Users speak a question, the app transcribes it (STT), forwards to a Cloudflare Worker that calls an LLM (OpenAI/Gemini/Grok/DeepSeek) using **user-provided keys**, then speaks back the answer (TTS). Extras: QR-based family onboarding, encrypted cloud backup of provider keys, high-contrast themes, JSON-driven onboarding.

1. High-Level Architecture

1. **Flutter app (Android-first)**
   - State: Riverpod
   - Nav: simple `Navigator` routes
   - Voice: `speech_to_text + flutter_tts`
   - Config/Themes/Onboarding: JSON assets (non-devs can edit)
2. **Cloudflare Worker**
   - KV storage for: encrypted key backups, family invites, family members
   - Endpoints: `/health`, `/api/chat`, `/api/keys/*`, `/api/family/*`
   - LLM proxy: calls provider API with the key sent from device (no server-side plaintext at rest)
3. **Providers (LLM APIs)**
   - OpenAI / Gemini / Grok / DeepSeek via HTTPS

2. Project Layout (essential paths)

```
/petitpal_mvp
  /petitpal                       # Flutter app
    /lib
      /config                     # central toggles & strings
      /src
        /analytics                # analytics wiring
        /family                   # invite/accept/list
        /home                     # landing
        /onboarding               # first-run flow
        /providers                # keys UI & Riverpod providers
        /security                 # AES-GCM keystore
      /theme                  # theme loader + preview
      /voice                  # STT/TTS + chat screen
    /assets
      /config                 # onboarding & provider labels
      /themes                 # JSON themes
  /cloudflare-worker          # worker.js + wrangler.toml
  /docs                       # setup, API spec, troubleshooting
```

3. Configuration (read first)

1. `lib/config/internal_config.dart`
   - `workerBaseUrl` : point to your Worker URL
   - Diagnostic toggles (analytics/crashlytics default off)
2. `lib/config/launch_config.dart`
   - Flip `LAUNCH_MODE = true` when shipping with Firebase configured
3. `lib/config/strings_config.dart`
   - All user-facing copy centralized
4. **Assets to tweak without code:**
   - `assets/themes/themes.json` (colors, tokens)
   - `assets/config/onboarding.json` (steps text)
   - `assets/config/provider_setup.json` (labels/help URLs)

4. App Bootstrap & Navigation

1. `lib/main.dart`
   - Loads default **high-contrast dark** theme via `ThemeLoader`
   - Wraps app in `ProviderScope` (Riverpod root)
2. `lib/app_router.dart`
   - Reads `isFirstRunProvider`
   - Routes:
     - `/onboarding → OnboardingScreen`
     - `/home → HomeScreen`
     - `/voice → VoiceScreen`
     - `/providers → ProviderSetupScreen`
     - `/family → FamilyHubScreen` (+ `/invite` , `/accept` , `/dashboard`)
     - `/themes → ThemePreviewScreen`
3. **First-run flag**

- `isFirstRunProvider` (SharedPreferences: `seen_onboarding`)

5. Theme & Accessibility

1. `lib/src/theme/registry.dart`
   - Loads a theme by ID from `assets/themes/themes.json`
   - Exposes `ThemeLoader.load(themeId, brightness)`
   - `PetitTokens` extension carries motion & corner radius
2. `lib/src/theme/theme_preview_screen.dart`
   - Quick visual check; change default theme by editing `main.dart` load call
3. **How to add a theme**
   - Add an object under `themes[]` in `assets/themes/themes.json`
   - Include `colors.dark|light` and `tokens.corner_radius/motion`

6. State & Storage (Riverpod)

1. `lib/src/providers/providers.dart`
   - `_prefsProvider` : `SharedPreferences` singleton
   - `deviceIdProvider` : stable UUID persisted in prefs
   - `isFirstRunProvider` : onboarding gate
   - `secureStorageProvider` : `FlutterSecureStorage`
   - `providerKeysProvider` : reads stored API keys (OpenAI/Gemini/Grok/DeepSeek)
2. **Key rules**
   - Keys are **stored locally** (Secure Storage)
   - Optional **encrypted backup** uploaded to Worker (see §8)

7. Voice Assistant Flow

1. **Files**
   - `lib/src/voice/voice_controller.dart` : STT/TTS state machine (`VoiceState`)
   - `lib/src/voice/voice_screen.dart` : UI for mic, transcript preview, send to backend
2. **Flow (sequence)**
   1. User taps **Start** → STT listens; interim transcript shown
   2. User taps **Stop** → state→ `processing`
   3. App resolves selected provider + reads key (Secure Storage)
   4. Calls `WorkerApi.chat(...)` with text + provider + key
   5. Receives response → display → TTS speak → state→ `idle`
3. **Backend call**

- `lib/src/worker_api.dart::chat()`
- POST `/api/chat` with headers: `X-Device-ID` and JSON body

4. **Error handling**
   - UI snackbar on exceptions; ensure at least one key is set

8. Provider Keys & Encrypted Backup

1. **Files**
   - `lib/src/providers/provider_setup_screen.dart` : key entry + backup
   - `lib/src/security/keystore.dart` : AES-GCM-256 + PBKDF2-HMAC-SHA256
2. **Local storage**
   - Keys saved under Secure Storage: `key_openai` , `key_gemini` , `key_grok` , `key_deepseek`
3. **Encrypted backup (optional but built-in)**
   - User enters a **backup password** (never leaves device)
   - `Keystore.encrypt(password, keysMap)` → ciphertext+nonce+salt
   - `WorkerApi.saveEncryptedKeys(deviceId, encrypted)` → KV put at `keys:<deviceId>`
4. **Server guarantees**
   - Worker stores **only encrypted blobs** (no plaintext keys)
   - To restore in a future version, client would fetch `/api/keys/get` and call `Keystore.decrypt(...)`

9. Family Onboarding (QR/Deep Link)

1. **Files**
   - `lib/src/family/invite_screen.dart` → create invite & show QR
   - `lib/src/family/accept_invite_screen.dart` → scan QR & join
   - `lib/src/family/family_dashboard_screen.dart` → list family members
   - Android deep-link intent is declared in `AndroidManifest.xml`
2. **Data model (KV keys)**
   - `family_by_owner:<deviceId>` → familyId (UUID)
   - `family:<familyId>` → `{ members: [{ device_id, name }], created_at, owner_device_id }`
   - `invites:<token>` (TTL 24h) → `{ family_id, member_name, issued_at }`
3. **Endpoints**
   - `POST /api/family/create_invite` → `{ family_id, invite_token, deeplink }`
   - `POST /api/family/accept_invite` → `{ family_id, member_name }` (adds member, deletes invite)
   - `GET /api/family/list` (header `X-Family-ID`) → `{ family_id, members: [...] }`
4. **Deep link**

- Worker issues a link like `https://<worker>.workers.dev/accept?token=...`
- Accept screen scans QR (contains the URL), extracts `token`, posts to Worker

10. Backend (Cloudflare Worker)

1. **File**: `cloudflare-worker/worker.js`
2. **Cross-cutting concerns**
   - CORS wide open for app: `Access-Control-Allow-Origin: *`
   - All JSON; explicit error responses `{ error: "..."}`
   - Required headers: `X-Device-ID` (chat/keys) or `X-Family-ID` (list)
3. **Endpoints**
   1. `GET /health` → `{ ok: true, version }`
   2. `POST /api/keys/save` → store encrypted backup at `keys:<deviceId>`
   3. `GET /api/keys/get` → return encrypted backup (404 if none)
   4. `POST /api/chat` → proxy to **one** provider based on `provider_hint`
      - OpenAI: `gpt-4o-mini`
      - Gemini: `gemini-1.5-pro:generateContent`
      - Grok: `grok-2-latest`
      - DeepSeek: `deepseek-chat`
   5. `POST /api/family/create_invite` → idempotently ensures a family for owner; creates TTL invite
   6. `POST /api/family/accept_invite` → validates token, appends member
   7. `GET /api/family/list` → returns members for a family
4. **KV layout**
   - See §9.2 for keys; TTL applied to invites (24h)
5. **Supabase-ready**
   - Storage surface is isolated: *only* `env["petitpal-kv"].get/put/delete` calls need swapping to Supabase queries later.
   - No change to client request/response contracts.

11. Networking (client)

1. **File**: `lib/src/worker_api.dart`
   - Centralizes all HTTP calls and headers
   - Throws on non-2xx → UI surfaces a snackbar
   - Timeout constant: `ApiConfig.requestTimeoutSeconds`
2. **Headers**
   - `User-Agent` set via `InternalConfig.appUserAgent`

- Device identity: `X-Device-ID` (from Riverpod provider)

12. Onboarding (first run)

1. **File**: `lib/src/onboarding/onboarding_screen.dart`
    - Loads steps from `assets/config/onboarding.json`
    - On finish, sets `seen_onboarding=true` → routes to `/home`
2. **Non-dev edits**
    - Change step titles/bodies in JSON, rebuild app

13. Analytics & Diagnostics

1. **Files**
    - `lib/src/analytics/events.dart` (canonical names)
    - `lib/src/analytics/analytics.dart` (Firebase init + `log`)
2. **Enable**
    - Drop `google-services.json` into `android/app/`
    - Set `LaunchConfig.LAUNCH_MODE = true`
    - (Crashlytics & Analytics toggle on automatically)

14. Build & Compatibility

1. **Known-good versions**: see `BUILD_CONFIG.md` (Flutter 3.22.2, AGP 8.4.2, Kotlin 1.9.24, SDK 35, minSdk 24, JDK 17, NDK 27.0.12077973)
2. **Script**: `scripts/compat_check.sh` prints env versions
3. **Run**
    - `flutter pub get`
    - `flutter run` on Android 8.0+

15. Security Model (keys & privacy)

1. **On device**: keys in **Secure Storage**
2. **In transit**: HTTPS only
3. **At rest (server)**: **encrypted blob** only (AES-GCM + PBKDF2)
4. **Password**: never uploaded; needed only to decrypt on a client

16. Future-Proofing & Supabase Migration (< 5 file changes)

1. **Worker: storage surface**
   - Replace `env["petitpal-kv"].get/put/delete` with Supabase calls (e.g., `families`, `invites`, `device_key_backups` tables).
   - Keep JSON shapes stable.
2. **Worker config**
   - Add Supabase URL/anon key as Worker Vars/Secrets; remove KV binding.
3. **Client**
   - No API change needed (same endpoints & payloads).
   - Optionally flip `InternalConfig.workerBaseUrl` if deploying a new Worker.

17. Extension Points (fast edits)

1. **Add an LLM provider**
   - Worker: add a `chatViaProvider("new", key, text)` branch
   - App: add dropdown item in `VoiceScreen` and save the key on Provider Setup
2. **Change copy**
   - Edit `strings_config.dart + assets/config/*.json`
3. **Add a page**
   - Create `lib/src/<feature>/<feature>_screen.dart`
   - Register a route in `app_router.dart`
4. **New theme**
   - Add to `assets/themes/themes.json`, use `ThemePreview` to test

18. File-by-File Quick Reference (numbered)

1. `lib/main.dart` — app bootstrap + theme load
2. `lib/app_router.dart` — routes + first-run gate
3. `lib/config/internal_config.dart` — backend URL & toggles
4. `lib/config/launch_config.dart` — launch/analytics switch
5. `lib/config/strings_config.dart` — user-facing text
6. `lib/src/voice/voice_controller.dart` — STT/TTS state & actions
7. `lib/src/voice/voice_screen.dart` — mic UI, provider select, send/receive
8. `lib/src/providers/providers.dart` — device ID, first-run, Secure Storage, keys accessor
9. `lib/src/providers/provider_setup_screen.dart` — enter & **encrypt-backup** keys
10. `lib/src/security/keystore.dart` — AES-GCM + PBKDF2 helpers
11. `lib/src/worker_api.dart` — all HTTP calls (health/keys/chat/family)
12. `lib/src/family/invite_screen.dart` — create invite + QR

13. `lib/src/family/accept_invite_screen.dart` — scan QR & join
14. `lib/src/family/family_dashboard_screen.dart` — list members by family ID
15. `lib/src/theme/registry.dart` — JSON→ThemeData + tokens
16. `lib/src/theme/theme_preview_screen.dart` — interactive theme checker
17. `lib/src/onboarding/onboarding_screen.dart` — JSON-driven onboarding
18. `assets/themes/themes.json` — colors/tokens (dark-high-contrast default)
19. `assets/config/onboarding.json` — steps copy
20. `assets/config/provider_setup.json` — provider labels/links
21. `cloudflare-worker/worker.js` — all endpoints + provider proxy
22. `cloudflare-worker/wrangler.toml` — KV binding & worker settings
23. `docs/*` — setup, API, troubleshooting, checklist