RecoDeck - One-Page App Summary

(Based only on repository evidence)

WHAT IT IS

RecoDeck is a desktop music library app built with Tauri, React, TypeScript, and Rust. It scans local audio folders, stores metadata in SQLite, supports playback, analyzes BPM/key, and adds AI-assisted chat/playlist generation.

WHO IT IS FOR

Primary persona (inferred from code and naming): digital DJs managing large local music libraries and playlist prep workflows.
Explicit persona statement in docs: Not found in repo.

WHAT IT DOES

- Scans library folders recursively and imports supported audio files (mp3, flac, wav, aiff/aif, m4a, ogg) with tag metadata and file hashes.
- Maintains track library data in SQLite (tracks, analysis, playlists, settings, genres) with migrations and indexed queries.
- Provides folder-tree and playlist organization, including playlist folders, add/remove tracks, rename/delete, and track counts.
- Plays audio with queue, next/previous, seek, repeat/shuffle, volume, and crossfade support, plus native decode fallback for unsupported formats.
- Analyzes tracks for BPM and musical key (Camelot/Open Key), stores results, and shows analysis progress for single-track, folder, or full-library runs.
- Watches library folders for file changes and emits "library-changed" events so the UI can re-scan and refresh automatically.
- Includes AI chat and AI playlist generation (Claude API key in settings), with cached library context rebuilt after library/analysis updates.

HOW IT WORKS (COMPACT ARCHITECTURE)

Components/services:

- Frontend: React app (`src/App.tsx`) + UI components + Zustand stores.
- Bridge: `src/lib/tauri-api.ts` calls Rust commands via Tauri `invoke`.
- Backend: Rust command modules (`src-tauri/src/commands/*`) for library, scanner, analysis, playback, playlists, genres, settings, watcher, and AI.
- Data layer: SQLite via rusqlite (`src-tauri/src/db/*`), DB file initialized at app data path as `recodeck.db`.

Data flow:

- User action in UI -> Tauri command -> Rust service -> SQLite/file system.
- Rust emits events (`library-changed`, `audio-chunk`, `audio-ended`, `audio-error`) -> frontend listeners/stores -> UI updates.

HOW TO RUN (MINIMAL GETTING STARTED)

1. Install JavaScript dependencies:
   npm install
2. Start the desktop app in development mode:
   npm run tauri dev
3. In the app UI, click "Scan Folder" (or add folders in Settings) to import your local music library.

Prerequisite version matrix (Node/Rust/toolchain versions): Not found in repo.