# Talkie — Feature Overview for Professionals

This document summarizes Talkie’s capabilities for clinicians, speech-language pathologists, care teams, and other professionals who work with people with speech impairments (e.g., dysarthria, Parkinson’s disease). It is informational rather than promotional.

## Feature list

- Local speech-to-text (Whisper or Vosk); continuous microphone capture with configurable sensitivity
- Local LLM (e.g., Mistral via Ollama) for sentence completion and clarification from partial or fragmented speech
- High-contrast, large-text UI with live status, volume display, and full history of transcriptions and responses
- User context and language profile (corrections, accepted responses, training facts) for personalized phrasing
- Edit past responses in History; “Use for learning” and scheduled curation to prioritize what the system learns
- Audio training facts (Train button): record short facts (names, relationships) injected into LLM context
- Document upload (TXT, PDF); vector store (Chroma); “Ask documents” mode to query by voice
- Voice-controlled web (optional): search, open URL, and store current page for RAG—all by voice
- Voice calibration and configurable chunk duration to tune for the user’s speech pattern
- Optional TTS (text-to-speech) for spoken responses
- All processing local; SQLite storage; export to JSONL for fine-tuning

## Overview

Talkie is a desktop application (laptop or Raspberry Pi) that helps people with reduced speech clarity communicate in conversation. The user speaks into a microphone; the system transcribes speech, completes or clarifies the intended sentence using a local language model, and displays the result in large, high-contrast text. Optionally, responses can be read aloud with text-to-speech. **All processing runs locally**—no speech or text is sent to the cloud.

---

## Core Capabilities

### Speech input and recognition

- **Continuous microphone capture** from a connected microphone, with configurable sensitivity (gain) to support quiet or reduced-volume speech.
- **Local speech-to-text (STT)** using either:
  - **Whisper** (default): Strong accuracy for dysarthric and speech-impaired speech; used in research for impaired speech. Configurable model size (e.g., small for speed, medium for best accuracy; medium needs more RAM).
  - **Vosk**: Lighter and faster; suitable for resource-limited devices (e.g., Raspberry Pi).

### Sentence completion and clarification

- **Local LLM (e.g., Mistral via Ollama)** turns partial or fragmented transcriptions into full, natural sentences as the user would say them (first person; requests as “Pass me the salt,” not “The user wants salt”).
- **Two-step flow (optional)**:
  - First pass interprets raw STT (e.g., homophone correction: “hockey” → “I’m”).
  - Second pass can be skipped when the first pass is confident, reducing latency.
- Prompts are designed to avoid inventing sentences and to respond with “I didn’t catch that.” when the input is unintelligible.
- **Configurable minimum transcription length** so very short or noisy segments are not sent to the LLM, reducing spurious or repeated wrong phrases.

### User interface

- **High-contrast, large-text PyQt6 interface** suitable for accessibility.
- **Fullscreen or windowed**; configurable font sizes for status and for the main response area.
- **Live status**: Listening / Transcribing / Responding (and error state).
- **Volume display**: Waveform-style strip showing microphone input level to aid positioning and gain adjustment.
- **History**: Past transcriptions and system responses stored and viewable in-app.

---

## Personalization and Learning

### User context and language profile

- **User context** (e.g., “PhD, professor at Brown,” preferred names, roles) can be set in Settings and is included in the LLM system prompt so output matches the user’s identity and style.
- **Language profile** is built from:
  - User context (Settings),
  - Explicit **corrections** (user edits of past responses in History),
  - **Accepted** completions (marked in History),
  - Optional **training facts** (see below).
- Profile content is capped (e.g., correction/accepted limits) and used to tailor vocabulary and phrasing.

### Correction and curation

- **Edit past responses** in the History view; corrections feed into the language profile so the system learns preferred wording.
- **“Use for learning”** checkbox in History lets the user or caregiver choose which interactions contribute to the profile.
- **Scheduled curation** (in-app or via CLI/cron): Pattern recognition over stored interactions; corrections and recurring phrases get higher weight; low-value or very old entries can be excluded or pruned. Higher-weighted examples are preferred when building the profile.

### Audio training facts

- **Train (T)** button: Record short facts (e.g., “Star is my dog,” “Susan is my wife”) that are injected into the LLM context so the model can use the user’s own names and relationships in responses.

---

## Documents and RAG

- **Document upload** (TXT, PDF) via the Documents dialog. Documents are chunked, embedded with a local embedding model (e.g., Ollama), and stored in a local vector store (Chroma; embedded or optional server).
- **“Ask documents” mode**: When enabled, the user asks a question by voice; the app retrieves relevant chunks and the LLM answers using only that context. Useful for asking about personal documents (e.g., care plans, notes) or reference material.
- **RAG is optional**: When “Ask documents” is off, conversation uses only the main system prompt and profile; no extra retrieval latency.

---

## Web / voice-controlled browser

When the optional browser feature is enabled, the user can switch into **browse (web) mode** and control the web by voice alone.

- **Search**: Speak a search query (e.g., "search for weather today"); the app opens the configured search engine (e.g., Google) with that query in the system browser (e.g., Chrome).
- **Open URL**: Speak a URL or site name (e.g., "open example dot com"); the app opens the corresponding URL in the browser.
- **Store page for RAG**: After opening a page, the user can say "store this page" (or equivalent). The app fetches the page, extracts text, chunks and embeds it with the same RAG pipeline used for uploaded documents, and adds it to the vector store under a source derived from the URL. Stored pages can then be queried via "Ask documents."
- **Configurable**: Search engine URL, browser app (e.g., Google Chrome on macOS), fetch

timeouts and retries, and an optional cooldown between actions can be set in `config.yaml`. The feature can be disabled with `browser.enabled: false`.

Browse mode is voice-only: the user toggles it on, then speaks browse commands (search, open URL, store page). All browse actions are executed locally; only page fetch goes over the network when the user requests a URL.

---

## Accessibility and Calibration

- **Voice calibration**: Record the user's speech and analyze level (and optionally STT/LLM behavior) to suggest microphone sensitivity and related settings.
- **Configurable chunk duration**: Longer capture windows allow more pause between words; shorter windows yield faster first response. Tunable for the individual's speech pattern.
- **TTS (text-to-speech)**: Optional spoken output of the system's response (e.g., macOS "say" with configurable voice).

---

## Data and Deployment

- **Local-only processing**: Speech and text stay on the device (or on-premises Ollama/Chroma if used). No cloud STT or LLM by default.
- **SQLite database**: Transcriptions, LLM responses, corrections, and training facts are stored locally (path configurable).
- **Export for fine-tuning**: Curated interactions can be exported to JSONL (instruction/input/output) for use with Ollama, Unsloth, or other tools to fine-tune a model on the user's patterns—useful for research or further personalization.

---

## Technical Requirements (summary)

- Python 3.11+

- Microphone
- Ollama running locally with a suitable LLM (e.g., Mistral)
- STT: Whisper (recommended for accuracy) or Vosk (lighter)
- Optional: Embedding model and Chroma for document RAG
- Optional: Browser feature for voice-controlled search, open URL, and store page for RAG (`browser.enabled` in config)

---

## Highlights for Clinical and Care Contexts

| Area | Highlight |
|---|---|
| **Privacy** | All processing local; no cloud dependency for core speech or LLM. |
| **Accuracy** | Whisper is a common choice in research for dysarthric speech; model size can be increased for best accuracy. |
| **Personalization** | User context, corrections, accepted responses, and training facts let the system adapt to the individual's vocabulary and style. |
| **Transparency** | Full history of what was heard and what was shown; corrections and "use for learning" give user/caregiver control over what is learned. |
| **Documents** | Optional RAG lets users ask questions about their own documents by voice. |
| **Web** | Optional voice-controlled web: search, open URL, and store current page for RAG—all by voice. |
| **Deployment** | Runs on laptop or Raspberry Pi; configurable for different hardware and speech profiles. |

This overview reflects the application's current feature set. For setup, configuration options, and troubleshooting, see the main README and `config.yaml`.