Voice Notepad

User Manual v3

Version 1.9.11 · December 2025

DUAL-PIPELINE ARCHITECTURE

Local Preprocessing + Cloud Transcription

PLATFORMS

Linux Debian/Ubuntu Wayland

Author: Daniel Rosehill
Repository: github.com/danielrosehill/Voice-Notepad

Table of Contents

AI-Human Co-Authorship

This software was developed through **AI-human collaboration**.

The code was generated by **Claude Opus 4.5** and other Anthropic models under my direction and supervision. I designed the architecture, specified requirements, reviewed outputs, and guided the implementation. Claude wrote the code.

This represents an experimental approach to software development. I believe it produces high-quality results when the human provides clear direction, domain expertise, and ongoing oversight.

The core design philosophy—**audio multimodal first**—came from my experience with traditional ASR-then-LLM workflows and recognizing that multimodal models could do both in a single pass. The iterative development process has been guided by my own daily use of the application.

— Daniel Rosehill, December 2025

Introduction

Voice Notepad is an **audio multimodal first** desktop application for voice transcription. The core design philosophy is to send audio directly to multimodal AI models that can "hear" and process audio alongside text instructions—rather than traditional ASR-then-LLM approaches.

1.1 Design Philosophy

This is an **experimental, iterative software design process** based on my own need for a better transcription tool. In the first few weeks of use, the approach has been validated with:

- **Over 2,000 transcriptions** processed
- **More than 1 million characters** of output
- Excellent accuracy and formatting results
- API costs of just a few dollars total

The key insight is that multimodal models can do both transcription and cleanup in a single pass, while also "hearing" tone, emphasis, and verbal commands that get lost in text-only processing.

1.2 What Makes This Different

Traditional Approach

1. Record audio
2. Send to ASR (Whisper, etc.)
3. Get raw transcript
4. Send text to LLM for cleanup
5. Get formatted output

Two API calls, higher cost

Voice Notepad Approach

1. Record audio
2. Local preprocessing (VAD + AGC)
3. Send audio + prompt to Gemini
4. Get formatted output

Single API call, lower cost
AI "hears" your voice

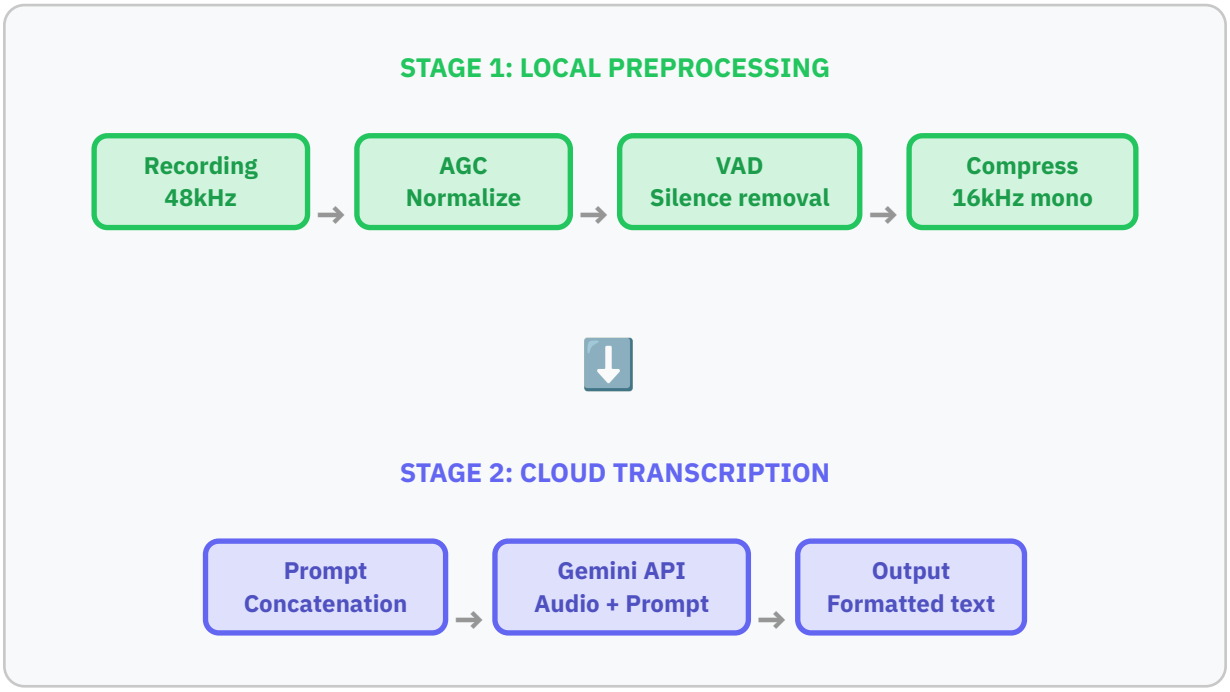
1.3 Key Features

- Dual-pipeline processing
- Voice Activity Detection (VAD)
- Automatic Gain Control (AGC)
- Single-pass multimodal transcription
- Layered prompt concatenation
- Configurable hotkeys (F13-F24)
- Text injection (auto-paste)
- Audio feedback (beeps or TTS)
- Transcript history with search
- Usage analytics and charts
- Cost tracking
- Statistics export

The Dual-Pipeline Architecture

Voice Notepad's effectiveness comes from combining local preprocessing with cloud-based multimodal transcription.

2.1 Pipeline Overview



The combination achieves:

- **Cost reduction**: VAD removes 30-80% of audio (silence/pauses)
- **Quality improvement**: AGC ensures consistent input levels
- **Precise formatting**: Layered prompts produce exactly the output you need
- **Single API call**: Multimodal models transcribe and format in one pass

2.2 Stage 1: Local Preprocessing

The local pipeline processes audio on your machine before any cloud upload.

1.1 Recording (PyAudio)

- Captures at device's native sample rate (typically 48kHz)
- 16-bit PCM, mono audio
- Automatic sample rate negotiation

- Handles microphone disconnection gracefully

1.2 Automatic Gain Control (AGC)

Normalizes audio levels for consistent transcription accuracy.

Parameter	Value	Purpose
Target peak	−3 dBFS	Optimal level with headroom
Min threshold	−40 dBFS	Skip if quieter than noise floor
Max gain	+20 dB	Prevent over-amplification

Behavior: Only boosts quiet audio—never attenuates loud audio.

1.3 Voice Activity Detection (VAD)

Removes silence using TEN VAD before API upload.

Parameter	Value	Purpose
Sample rate	16kHz	Required by TEN VAD
Hop size	256 samples	16ms analysis windows
Threshold	0.5	Speech probability cutoff
Min speech	250ms	Ignore very short sounds
Padding	30ms	Buffer around speech

Typical reduction: 30-80% depending on speaking pattern.

1.4 Compression

- Downsampled to 16kHz mono (matches Gemini's internal format)
- Converted to WAV for API upload
- 66% smaller than 48kHz stereo input

2.3 Stage 2: Cloud Transcription

2.1 Prompt Concatenation

Instructions are built from multiple layers:

Foundation Layer (always applied):
- Remove filler words (um, uh, like, you know)
- Add punctuation and paragraph breaks
- Follow verbal commands (“scratch that”, “new paragraph”)
- Fix grammar and spelling

Format Layer (based on preset):
- Email, meeting notes, bullet points, documentation, etc.

Style Layer:
- Formality: casual, neutral, professional
- Verbosity: none to maximum reduction

Personalization:
- Email signatures injected for email format

2.2 Multimodal API Submission

- Audio is base64-encoded with concatenated prompt
- Gemini processes audio + text instructions together
- The AI “hears” tone, emphasis, and verbal commands
- Returns formatted text in a single API call

Installation

Voice Notepad requires a two-stage installation: system dependencies first, then the application.

3.1 Stage 1: System Dependencies

Install required system packages on Ubuntu/Debian:

```
sudo apt install python3 python3-venv ffmpeg portaudio19-dev libc++1
```

Package	Purpose
`python3`, `python3-venv`	Python runtime and virtual environments
`ffmpeg`	Audio format conversion and compression
`portaudio19-dev`	PyAudio recording library headers
`libc++1`	Required by TEN VAD for voice activity detection

3.2 Stage 2: Application Installation

3.2.1 Option A: Debian Package (Recommended)

```
sudo apt install ./voice-notepad_1.9.11_amd64.deb
```

3.2.2 Option B: AppImage

```
chmod +x Voice_Notepad-1.9.11-x86_64.AppImage
./Voice_Notepad-1.9.11-x86_64.AppImage
```

3.2.3 Option C: From Source

```
git clone https://github.com/danielrosehill/Voice-Notepad.git
cd Voice-Notepad
./run.sh
```

Hardware Recommendations

Voice Notepad is designed to work with dedicated hardware for hands-free operation.

4.1 Design Rationale: F13-F24 Keys

The application uses **F13 through F24** as the default hotkey range for an important reason: **these keys are defined in Linux but virtually never used by applications**.

This design choice ensures:
- **No conflicts** with user-level programs (browsers, editors, etc.)
- **No conflicts** with desktop environment shortcuts
- **System-level interception** without interfering with normal keyboard use
- **Clean separation** between transcription controls and regular typing

Most standard keyboards only have F1-F12. To use Voice Notepad's hotkeys, you need either a keyboard with extended function keys or a separate input device mapped to F13-F24.

4.2 Recommended Hardware

4.2.1 Simple: USB HID Button ($5)

A single USB HID programmable button (available on AliExpress for $5) provides excellent **push-to-talk (PTT)** functionality.

- Map the button to **F15** (Toggle Recording)
- Press to start recording, press again to transcribe
- Simple, reliable, inexpensive
- No software required after initial mapping

4.2.2 Full Setup: Macro Pad

For power users, a USB macro pad with 6+ keys enables the full hotkey workflow:

Button	Maps To	Function
1	F15	Toggle (start/stop and transcribe)
2	F16	Tap Toggle (start/stop and cache)
3	F17	Transcribe cached audio
4	F18	Clear recording

Button	Maps To	Function
5	F19	Append to cache
6	F20	Pause/resume

The number of configurable hotkeys (6 functions) is specifically designed to match typical macro pad layouts.

4.2.3 Alternative: Foot Pedal

USB foot pedals are popular for hands-free transcription:

- Keep hands on keyboard while controlling recording
- Map pedal buttons to F15/F17/F18
- Available from $15-30

4.3 Setting Up Key Mapping

Use **Input Remapper** on Linux to map your device:

```
sudo apt install input-remapper
```

1. Open Input Remapper
2. Select your USB device
3. Click “Record” and press the button
4. Set output to `KEY_F15` (or desired F-key)
5. Click Apply and enable Autoload

Configurable Hotkeys

5.1 Default Mappings

Key	Function	Description
F15	Toggle	Start recording, or stop and transcribe immediately
F16	Tap Toggle	Start recording, or stop and cache
F17	Transcribe	Transcribe cached audio
F18	Clear	Delete recording and clear cache
F19	Append	Start recording that appends to cache
F20	Pause	Pause/resume current recording

5.2 Configuration

Go to **Settings → Hotkeys** to customize mappings. Each function can be assigned any key from F13-F24, or disabled entirely.

5.3 Technical Notes

- On Linux/Wayland: Hotkeys work via evdev (reads directly from input devices)
- Requires user to be in the `input` group: `sudo usermod -aG input $USER`
- Falls back to pynput/X11 on non-Linux systems

Audio Feedback

Voice Notepad provides audio notifications for recording events.

6.1 Modes

Configure in **Settings → Behavior → Audio feedback**:

Mode	Description
Beeps	Short beep tones for events (default)
Voice (TTS)	Spoken announcements via Edge TTS
Silent	No audio feedback

6.2 TTS Announcements

When Voice mode is enabled, you'll hear spoken feedback:

Event	Announcement
Recording started	"Recording"
Recording stopped	"Recording stopped"
Audio cached	"Cached"
Transcription started	"Transcribing"
Transcription complete	"Complete"
Text copied	"Text on clipboard"
Recording cleared	"Recording discarded"
Error occurred	"Error"

TTS uses pre-generated audio files with a British English male voice (en-GB-RyanNeural via Microsoft Edge TTS).

Transcript History

All transcriptions are automatically saved to a local MongoDB-compatible database.

7.1 Database

Voice Notepad uses **Mongita**, a pure Python MongoDB implementation. Data is stored locally at `~/.config/voice-notepad-v3/mongita/`.

Each transcript record includes:
- Full transcript text
- Timestamp
- Provider and model used
- Audio duration (original and after VAD)
- Inference time
- Token usage and cost
- Word and character counts

7.2 History Tab Features

- **Full-text search** across all transcriptions
- **Click to preview** any transcript
- **Double-click to load** into editor for reuse
- **Delete individual** transcriptions
- **Delete all** with confirmation

Analytics & Cost Tracking

8.1 Analytics Tab

The Analytics tab provides performance insights:

- **Summary statistics**: Total transcriptions, words, characters
- **Daily activity chart**: Visual bar chart with metric toggles
- **Model performance table**: Compare inference times across models
- **Time period selector**: Today, 7 days, 30 days, or all time

8.2 Export Statistics

Click **Export Stats** to save anonymized statistics as JSON. Exported data includes:
- Transcription counts and volume
- Model performance metrics (no transcript content)
- Daily activity breakdown

Useful for benchmarking or sharing performance data.

8.3 Cost Tab

For OpenRouter users, the Cost tab shows:

- **Account balance**: Live credit balance
- **Key-specific usage**: Daily, weekly, monthly spend
- **Model breakdown**: Usage by model

8.4 Cost Effectiveness

Real usage data with Gemini 2.5 Flash:
- 848 transcriptions for $1.17 total
- 84,000 words transcribed and cleaned
- About $0.014 per 1,000 words (1.4 cents)

Text Injection (Wayland)

Text injection automatically pastes transcribed text at your cursor after transcription.

9.1 Requirements

- `ydotool` package installed
- `ydotoold` daemon running as your user (not root)

9.2 Quick Setup

```
sudo apt install ydotool
sudo pkill ydotoold
sudo rm -f /tmp/.ydotool_socket
ydotoold &
```

9.3 Verify

```
ls -la /tmp/.ydotool_socket
# Should show YOUR username as owner, not root
```

9.4 Persistent Setup

Create `~/.config/systemd/user/ydotoold.service`:

```
[Unit]
Description=ydotool daemon

[Service]
ExecStart=/usr/bin/ydotoold

[Install]
WantedBy=default.target
```

Enable: `systemctl --user enable --now ydotoold`

Output Modes

Three independent output modes can be combined:

Mode	Behavior
App	Text appears in the application window
Clipboard	Text copied to system clipboard
Inject	Text typed at cursor via ydotool

Enable any combination: all three, any two, or just one.

Storage Locations

All data is stored in `~/.config/voice-notepad-v3/`:

Path	Contents
`config.json`	API keys and preferences
`mongita/`	MongoDB-compatible transcript database
`usage/`	Daily cost tracking JSON files
`audio-archive/`	Opus audio recordings (if enabled)

Troubleshooting

12.1 Audio Issues

No microphone detected: Check `pactl list sources short`. Verify PipeWire is running.

Poor quality: Enable AGC in Settings → Behavior. Position microphone closer.

12.2 API Issues

Transcription fails: Verify API key. Check internet. Try different model.

High costs: Enable VAD. Use Flash Lite models.

12.3 Hotkeys Not Working

Add yourself to input group: `sudo usermod -aG input $USER` then log out/in.

12.4 Text Injection Not Working

Verify ydotoold is running as your user, not root. Check socket ownership.

Support

Voice Notepad is open source software provided on a **best-effort basis**. No formal support is provided.

- **Repository**: https://github.com/danielrosehill/Voice-Notepad
- **Documentation**: https://github.com/danielrosehill/Voice-Notepad/tree/main/docs

Bug reports and feature requests can be submitted via GitHub Issues, but responses are not guaranteed.

Voice Notepad User Manual v3
Version 1.9.11 · December 2025
Created by Daniel Rosehill with Claude (Anthropic)
MIT License