



WakkaQt

THE OFFICIAL UNOFFICIAL MANUAL

"Because Auto-Tune is expensive and your bathroom acoustics only get you so far." A field guide to plugging in a microphone, embarrassing yourself with confidence, and walking away with a suspiciously professional-looking video.

COVERS VERSION 2.7.7

No subscriptions. No cloud. No judgment. (The pitch monitor may disagree.)

Table of Contents

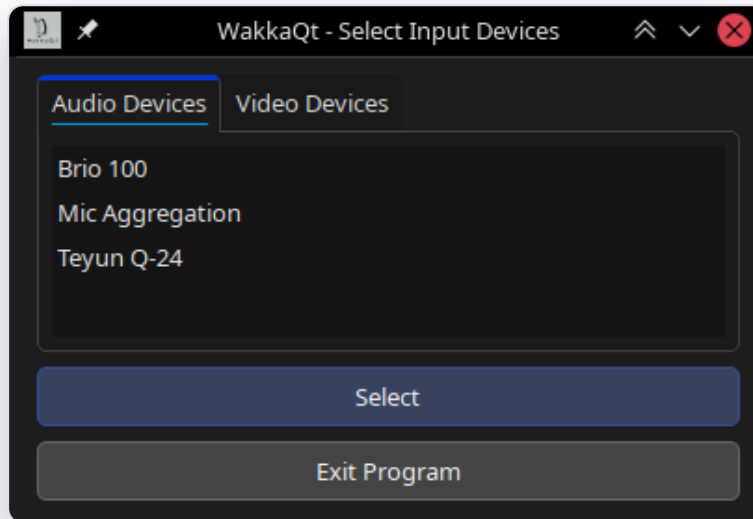
01	Choosing Your Weapons — the device dialog	4
02	The Main Stage — a tour of the window that judges you	5
03	Feeding the Beast — loading files & raiding YouTube	6
04	Showtime — pressing the big glowing SING button	8
05	The Karaoke Studio — tuning, syncing, and special effects	9
06	The Backing Track Sorcery — AI vocal removal	12
07	Rendering Your Masterpiece	13
08	The Session Library — your shame, archived forever	14
A	Appendix: The Full Feature Checklist	15
B	Appendix: Building on Linux (a feature, not a bug)	16

What is this thing? WakkaQt is a free, open-source karaoke recording and production studio built on Qt6. You load a karaoke video, sing into a microphone, and the software runs your voice through a genuinely serious digital signal processing pipeline — pitch correction, noise reduction, reverb, dynamics — before rendering a finished MP4 with your webcam and a pitch meter that exists purely to keep you honest. This manual walks through every screen, in the order you're most likely to actually encounter them, with real screenshots and an inadvisable number of jokes.

Choosing Your Weapons

In which you tell the software what it's allowed to listen to.

Every WakkaQt session begins with an interrogation. Before you're allowed anywhere near the main window, the app wants to know exactly which microphone and (optionally) which webcam it's permitted to use. This is a good thing: it means WakkaQt is never guessing, and it's never quietly recording out of the wrong USB headset while you serenade a device that isn't plugged in.



The Select Input Devices dialog. Two tabs — Audio Devices and Video Devices — list everything your operating system currently admits to having.

- 1 Click the **Audio Devices** tab and pick your microphone from the list. WakkaQt enumerates every input device your OS reports, so if you've got a USB condenser mic, an aggregated multi-mic setup, or just the laptop's built-in potato, it'll show up here.
- 2 Switch to the **Video Devices** tab if you want webcam footage in the final render. This step is entirely optional — skip it and WakkaQt happily runs audio-only, no camera required, no drama.
- 3 Press **Select** to confirm and open the main window. Or press **Exit Program** if you've had a change of heart about the whole "singing" idea. No judgment. Yet.

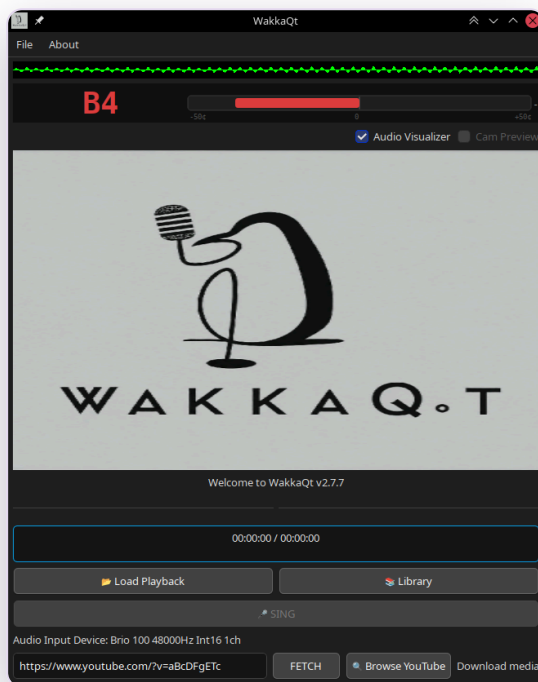
PRO TIP

Didn't plug in a webcam? Don't panic and don't unplug anything in a hurry — WakkaQt falls back to a clean audio-only pipeline end-to-end: recording, preview, and render all skip the camera gracefully. Nobody has to see you like this.

The Main Stage

A guided tour of the window you'll be staring at for the next several hours.

This is home base. Every recording, every render, and every regret starts and ends here. Let's look at what's actually on screen, top to bottom.



The main window at rest. Calm. Patient. Waiting.

- **Pitch monitor (top strip):** always visible, always listening, always ready to display an enormous letter like **B4** next to a cents bar that goes green, yellow, or red depending on how close you are to the note you were supposed to be singing. It's on even when you're not recording — free, unsolicited feedback.
- **Audio Visualizer toggle:** mutes or unmutes the twin waveform strips flanking the webcam preview, showing the song's own playback audio — not your microphone.
- **Video area:** shows the loaded karaoke video, or the WakkaQt logo if nothing is loaded yet, patiently waiting for you to feed it something.
- **Transport / status bar:** elapsed time, current song title once loaded, and the playback controls once media is on deck.
- **Load Playback** and **Library** buttons: open a file from disk, or reopen a previous session from your ever-growing personal archive of karaoke attempts.
- **SING** button: large, unmissable, mildly intimidating. Covered properly in Chapter 4.
- **Audio Input Device** line: a permanent reminder of which microphone you committed to back in Chapter 1, including its live sample rate and bit depth, for anyone who enjoys that sort of detail.
- **URL field + FETCH / Browse YouTube / Download media:** the fastest route from "I want to sing this song" to "I am now singing this song." More in Chapter 3.

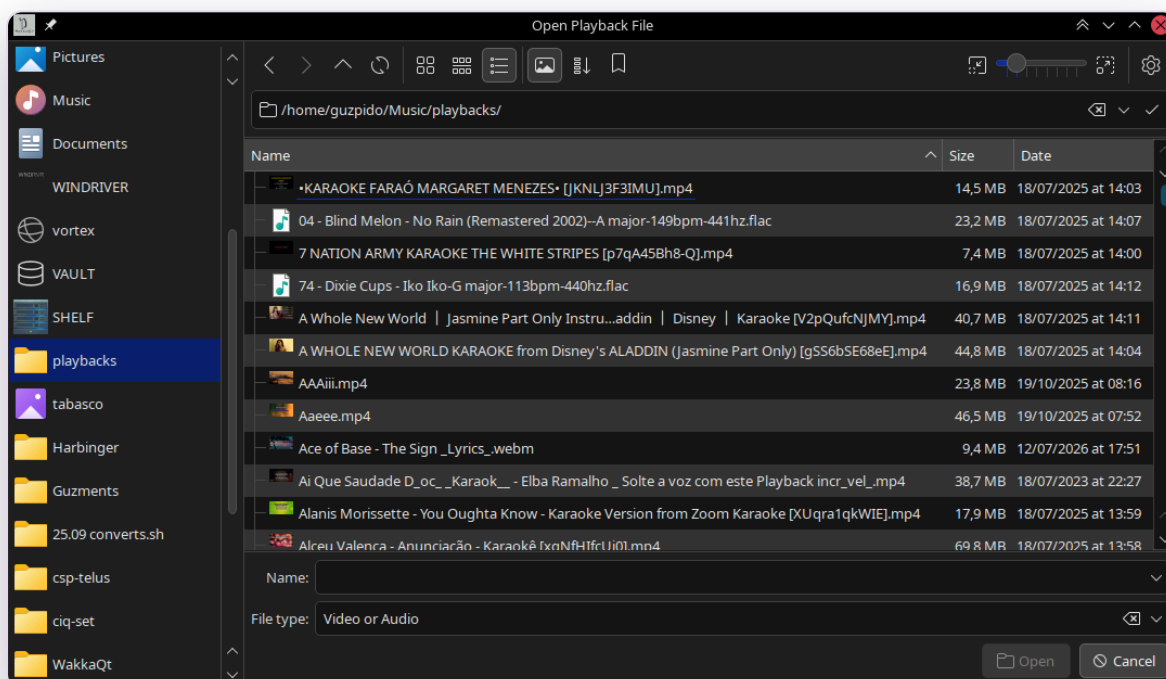
Feeding the Beast

Getting a song onto the screen, by whatever means necessary.

WakkaQt doesn't care where your karaoke track comes from, as long as it's a format Qt6 Multimedia understands: MP4, MKV, WebM, AVI, MOV, MP3, WAV, FLAC, or OPUS. There are two respectable ways to get one loaded, and one delightfully lazy third way.

Option A — Open a file you already have

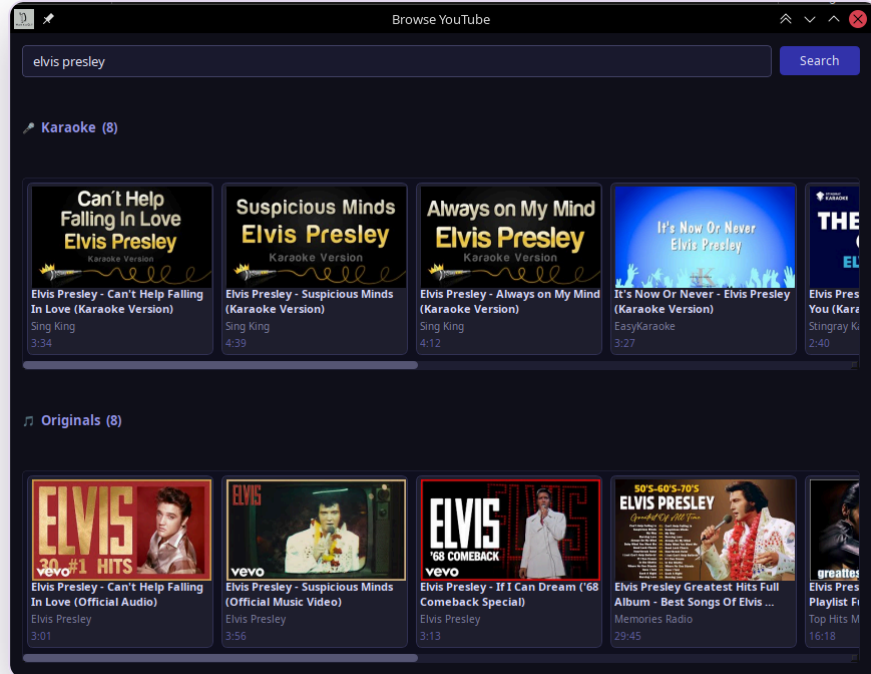
Click **Load Playback** and pick a file from your existing collection. This is the standard, no-surprises file picker your operating system already gave you — nothing to explain here except that, judging by the screenshot below, some people have a *lot* of karaoke files already.



The Open Playback File dialog. Yes, that is a folder full of hundreds of megabytes of other people's greatest hits, and no, we are not going to talk about it.

Option B — Browse YouTube from inside the app

Click **Browse YouTube** to open a proper search dialog with two scrollable rows of results — dedicated **Karaoke** versions on top, **Originals** below, in case you'd rather learn the song first before humiliating yourself with the backing track. Click a card for a preview, then "Download this video" to send it straight into the download flow (powered by `yt-dlp` under the hood).



Searching "elvis presley" turns up eight karaoke versions and eight originals. Ambitious choice for a Tuesday.

OPTION C — THE LAZY WAY

Already have a YouTube link on your clipboard? Paste it straight into the URL field on the main window and hit **FETCH**. Same `yt-dlp` download pipeline, zero browsing required.

Once something's loaded, the main window comes alive: the video plays, a waveform scrolls by, and the full transport bar (rewind, play/pause, stop, fast-forward) appears above the buttons.



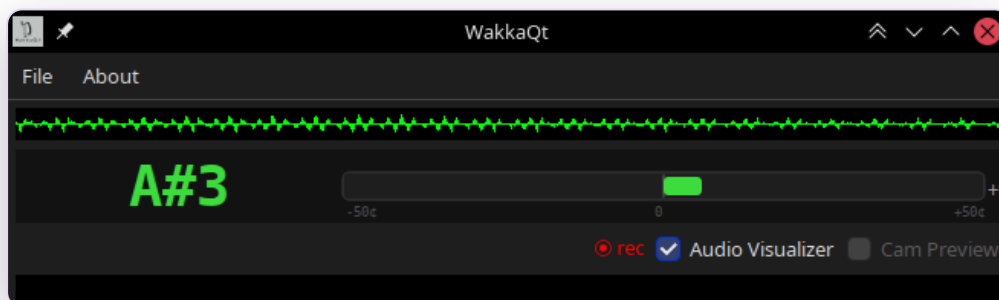
A track loaded and playing — album art, title, and a live waveform scrubber, with Load Playback, Load Again, Library, SING, and Backing Track all standing by.

Showtime

The button you've been building up to. Press it. We dare you.

With a song loaded and a device selected all the way back in Chapter 1, the  **SING** button springs to life. Press it, and here's exactly what happens:

- 1 Playback starts from the beginning (or wherever you last parked it).
- 2 Your microphone starts recording, live, uncompressed, at its native sample rate — no forced downsampling, so if your mic does 48 kHz, WakkaQt processes it at 48 kHz all the way through.
- 3 If you enabled a webcam back in Chapter 1, it records alongside the audio, synced to the same clock.
- 4 The pitch monitor strip — the one that's been quietly watching you this whole time — switches into recording mode: a red `rec` indicator appears, and the note name flips between green, yellow, and red in real time as you chase (or flee) the correct pitch.
- 5 Recording stops automatically when the song ends, or you can stop it manually at any point.



Mid-take: the rec indicator is lit, the note reads A#3, and the bar is sitting comfortably in the green. This is what going well looks like.

REALITY CHECK

Nothing is being auto-tuned live. The pitch monitor is purely informational during recording — a coach standing behind you with a clipboard. The actual correction happens afterward, in the Karaoke Studio, where you have full control over how much (or how little) of it to apply.

Once the song ends and the recording stops, WakkaQt automatically opens the **Karaoke Studio** — the preview and enhancement dialog — with your freshly recorded vocal already loaded and waiting.

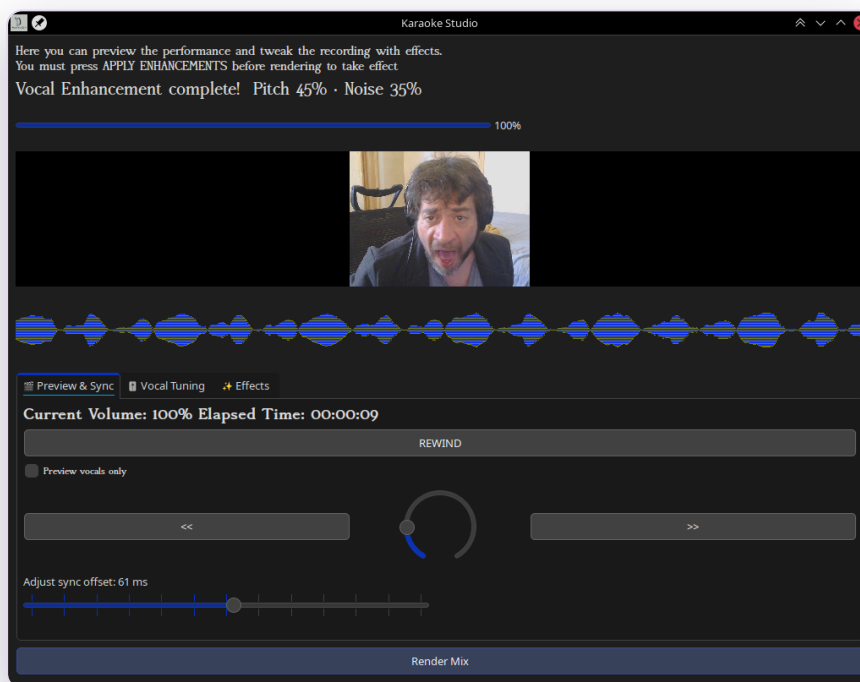
The Karaoke Studio

Where raw human effort quietly becomes something you'd admit to on social media.

The Karaoke Studio (technically the Preview dialog, but "Karaoke Studio" is what it proudly calls itself in the title bar) is where your vocal take gets processed, previewed, tweaked, and previewed again — as many times as your patience allows — before you commit to a final render. It's organized into three tabs, and none of them are optional to understand.

Tab 1 — Preview & Sync

The default tab. A live webcam preview (if you recorded one) sits above a scrolling audio visualizer, kept perfectly in sync with playback — rewind, seek, or nudge the offset, and the video resyncs immediately. Below that: transport controls, a "Preview vocals only" checkbox for isolating your voice from the backing track, seek buttons, and — critically — the **sync offset** slider, for the inevitable few milliseconds of drift between your voice and the video.



Preview & Sync — enhancement already applied (Pitch 45% · Noise 35%), sync offset dialed in at 61 ms.

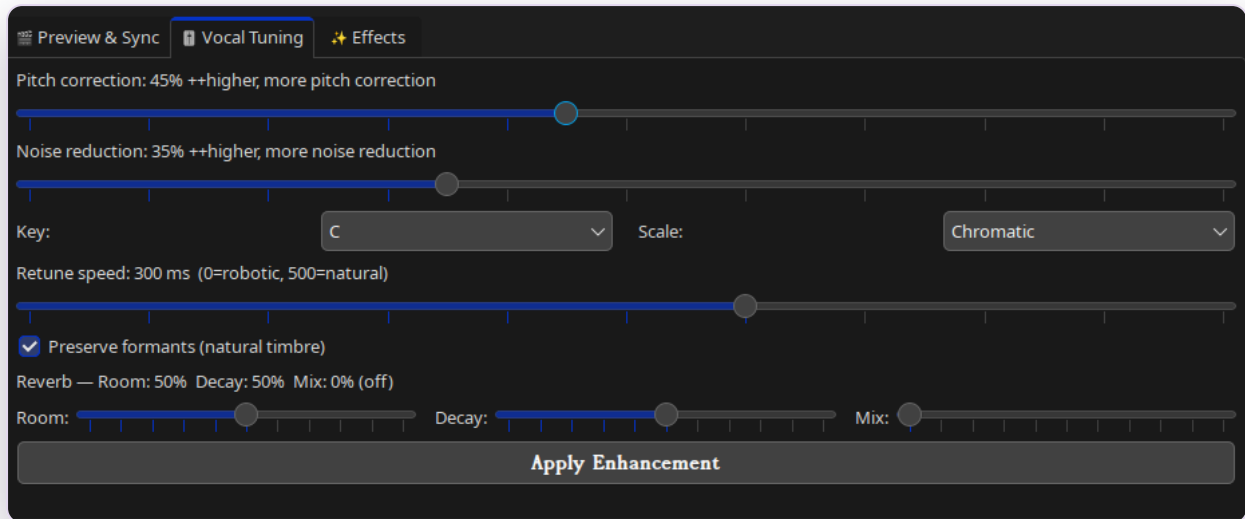
Tab 2 — Vocal Tuning

This is the actual DSP pipeline, exposed as sliders instead of intimidating audio-engineer jargon:

- **Pitch correction** — phase-vocoder pitch shifting. 0% leaves your voice completely alone; higher values snap harder toward the target note.
- **Noise reduction** — a spectral subtraction gate with adaptive noise floor estimation, for evicting fan hum, keyboard clatter, and the neighbor's leaf blower.
- **Key & Scale** — pick a key (any of the 12) and a scale (Major, Minor, Pentatonic, Blues, Dorian, Mixolydian, Lydian, Phrygian, Locrian, Harmonic Minor, Melodic Minor, Whole Tone, Diminished, or plain Chromatic if you'd rather not commit) so pitch correction only ever snaps to notes that actually belong in the song.
- **Retune speed** — 0 ms for the unmistakable, fully robotic T-Pain effect; up to 500 ms for a slow, natural glide that barely admits it's correcting anything.
- **Preserve formants** — an LPC-based envelope re-synthesis step that keeps your voice sounding like a human being even after aggressive pitch shifting, instead of a chipmunk with opinions.

- **Reverb** — Freeverb-style Schroeder reverb with independent Room, Decay, and Mix controls, for turning your bedroom into a cathedral, or at least a slightly larger bedroom.

Adjust anything, press **Apply Enhancement**, and the preview updates so you can hear exactly what you've done to yourself.



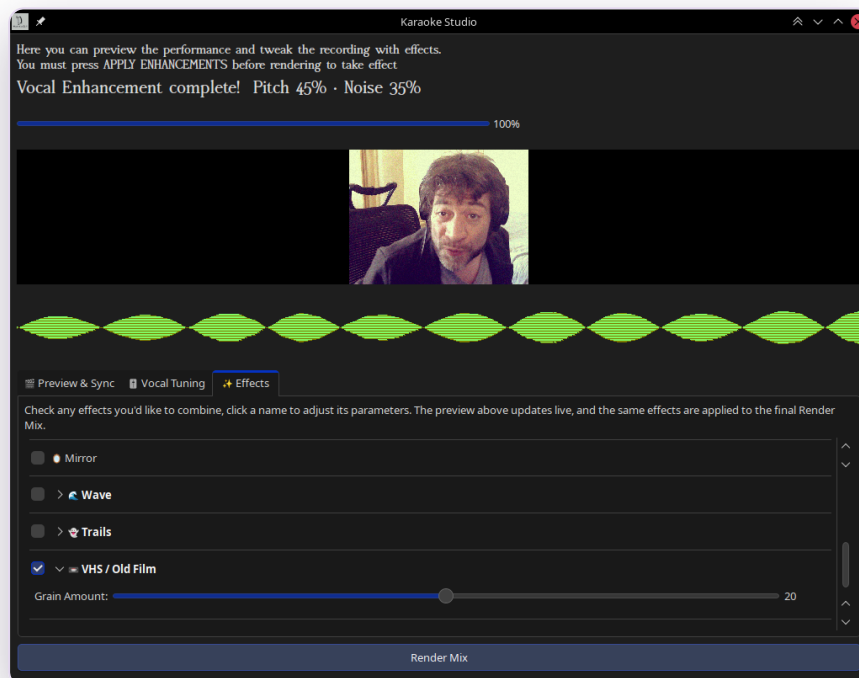
Vocal Tuning: Pitch correction 45%, Noise reduction 35%, key of C, Chromatic scale, retune speed 300 ms, formants preserved, reverb parked at 0% mix (i.e., off, for now).

Special Effects

Because a technically correct pitch isn't nearly enough spectacle.

Tab 3 — Effects

Twelve combinable video effects, each one a friendly slider or two standing in for a raw `libavfilter` chain you'll never need to type by hand:  Vertigo,  Technicolor,  Sepia,  High Contrast,  Black & White,  Vignette,  Negative,  Mirror,  Wave,  Trails,  VHS/Old Film, and  Edge Detect. Check as many as you like — they chain together in order — and the live preview above updates instantly, using the exact same filter chain that gets baked into the final render. What you see really is what you get.



The Effects tab, with VHS / Old Film checked and its Grain Amount slider cranked to 20 — because if you're going to relive your youth, you may as well look like a camcorder tape from it.

BONUS CONTENT

 Vertigo runs on a `frei0r` plugin instead of a plain filter. If that plugin isn't installed on your system, WakkaQt simply hides the option from the list rather than throwing an error at you — no crash, no drama, just one fewer way to make your webcam feed spiral menacingly.

Once everything sounds and looks right, press **Render Mix** at the bottom of the dialog to move on to Chapter 7. Or keep tweaking. The Studio will wait. It has nowhere else to be.

The Backing Track Sorcery

Turn any song into a karaoke track, using a small AI you own outright.

Loaded a normal song — vocals and all, no "instrumental" or "karaoke" label in sight — and want to sing over it anyway? Click  **Backing Track**, visible right below the SING button whenever a file is loaded.

- 1 On first use, WakkaQt downloads the **UVR-MDX-NET-Inst_HQ_3** ONNX vocal-separation model (about 80 MB), verifies it against a pinned checksum, and caches it in `~/WakkaQt/models/`. This happens exactly once, ever.
- 2 The model separates vocals from the instrumental using MDX-Net deep learning, run entirely on your own machine through a full STFT/iSTFT pipeline. No internet connection needed after that first download, no cloud service involved, nothing leaves your computer.
- 3 If the source was a video file, the separated instrumental is muxed straight back onto the original video by default, so you keep the visuals. Prefer audio only? Choose WAV or MP3 in the save dialog instead.
- 4 Changed your mind halfway through? The **Abort** button cancels separation cleanly, mid-flight, no corrupted output left behind.

WHY THIS MATTERS

This is the entire "no subscription, no cloud" philosophy in one feature. Plenty of tools charge you monthly, or upload your audio to a server somewhere, to do exactly this. WakkaQt just downloads a small model once and does the math locally, on your CPU, for free, forever.

Rendering Your Masterpiece

The moment your bathroom-acoustics performance becomes a permanent, shareable file.

Press **Render Mix** in the Karaoke Studio, and WakkaQt assembles everything into a finished 1920×1080 MP4:

- Only your webcam feed appears in the final video — the karaoke video's picture is never included, just its audio.
- Your vocal is mixed in with every enhancement from Chapter 5 fully applied — pitch correction, noise reduction, formant preservation, reverb, plus a final polish pass of compression, limiting, and harmonic excitement for a loud, clean master.
- A **pitch indicator strip** is burned directly into the webcam frame: the note name, a cents-deviation bar, and a `+/-N¢` value, color-coded green (in tune), yellow (drifting), or red (having a moment) — a permanent, rewatchable record of exactly how your pitch held up.
- Rendering runs natively in-process via FFmpeg's libraries with a real-time progress bar, and automatically tries hardware-accelerated H.264 encoding (NVENC, then VAAPI, then V4L2M2M) before falling back to software encoding — whichever your machine can actually do fastest.
- Changed your mind mid-render? **Abort Render** stops it cleanly without leaving a broken file behind.

NO FFmpeg DEV LIBRARIES AT BUILD TIME?

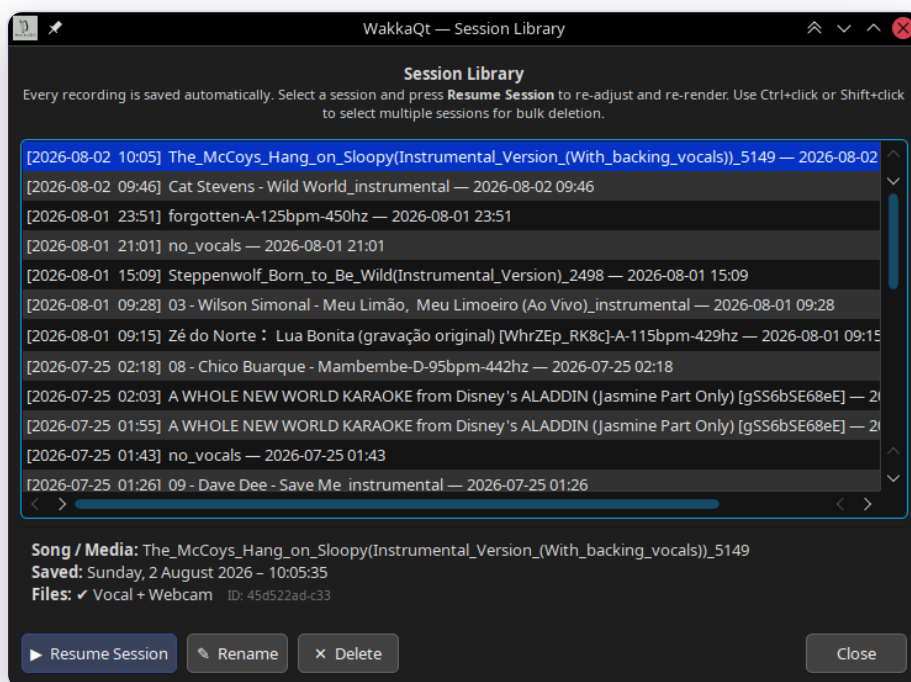
No problem. WakkaQt falls back gracefully to spawning a system `ffmpeg` process instead. Slightly less elegant under the hood, identical result on screen.

When it's done, you have a finished, shareable video — and, whether you asked for it or not, a permanent record of every note you tried to sneak past the pitch monitor.

The Session Library

Every recording, automatically archived, whether you like it or not.

Every time you record, WakkaQt quietly saves the whole session to `~/.WakkaQt/library/` — a unique folder, all the source files, and a JSON manifest describing what's inside. Click **Library** from the main window to see the full, timestamped history of your karaoke career.



The Session Library. A running, timestamped ledger of every song you've ever sung into this program, going back weeks. There is no "delete history and pretend this never happened" button, but there is a perfectly good Delete button for individual sessions.

- **Resume Session** — reload any past recording straight back into the Karaoke Studio, with all its original vocal and webcam files, ready to re-tune and re-render with completely different enhancement settings. The intermediate processed vocal is never stored — it's regenerated fresh from your raw take every time, so the preview always reflects your current sliders, not last week's.
- **Rename** — give a session a name you'll actually recognize later, instead of whatever the source filename happened to be.
- **Delete** — remove a session for good. Hold **Ctrl** or **Shift** while clicking to select and bulk-delete several at once, for the occasional deep clean.

SAVED TRANSACTIONALLY

Sessions are written to a temporary `.partial` folder first and only renamed into place once everything has saved successfully. A crash or a closed lid mid-save can't leave a broken, half-written entry cluttering your library.

The Full Feature Checklist

For skimmers, and for anyone who just wants the receipts.

Feature	Status
MP4 / MKV / WebM / MP3 / WAV / FLAC playback	✓ Yes
Microphone recording (selectable device)	✓ Yes
Webcam recording	✓ Yes
Audio-only recording (no webcam required)	✓ Yes
Real-time pitch monitor (YIN algorithm, always on)	✓ Yes
Real-time waveform visualizer	✓ Yes
Pitch correction (phase vocoder)	✓ Yes
Scale/key-aware pitch snapping	✓ Yes
Formant preservation (LPC)	✓ Yes
Noise reduction (spectral subtraction)	✓ Yes
Reverb (Freeverb / Schroeder)	✓ Yes
Compressor + limiter + harmonic exciter	✓ Yes
Preview dialog with live tweaking	✓ Yes
Native FFmpeg rendering (in-process)	✓ Yes
Pitch overlay burned into rendered video	✓ Yes
Live webcam video preview (synced)	✓ Yes
12 combinable video effects	✓ Yes
Session library (save / rename / delete / re-render)	✓ Yes
YouTube download (via yt-dlp)	✓ Yes
YouTube karaoke browser (search + preview)	✓ Yes
AI vocal separation → backing track (local ONNX model)	✓ Yes
Backing track: video-preserving output	✓ Yes
Hardware-accelerated H.264 (VA-API / NVENC)	✓ Yes
Abort render / abort separation	✓ Yes
Cross-platform (Linux / Windows)	✓ Yes
Subscription required	✗ No
Phones home to a server	✗ No
Judgment about your singing	mostly ✗ No

Building on Linux

"A feature, not a bug" — the official project stance.

Windows users get a ready-made binary. Linux users get character-building. Here's the abridged version — see the project README for the complete, distro-by-distro rundown.

Debian / Ubuntu prerequisites

```
sudo apt install build-essential cmake ninja-build qt6-base-dev qt6-multimedia-dev libfftw3-dev  
libavformat-dev libavcodec-dev libavfilter-dev libavutil-dev libswresample-dev libswscale-dev  
libglib2.0-dev pkg-config
```

Configure and build

```
cmake -B build -DCMAKE_BUILD_TYPE=Release  
cmake --build build --parallel  
./build/WakkaQt
```

STRONGLY RECOMMENDED

Install `libavcodec-extra` (Debian/Ubuntu) so recordings actually use H264 instead of falling back to MotionJPEG. And keep `ffmpeg` and `yt-dlp` on your `$PATH` at runtime — both are used even in the native-FFmpeg build, as fallbacks and for downloads respectively.

OPTIONAL: THE AI BACKING-TRACK FEATURE

Install ONNX Runtime (`libonnxruntime-dev` on Debian/Ubuntu) before configuring. It's entirely optional — without it, WakkaQt builds and runs normally, and the Backing Track button simply doesn't appear.

Full instructions, Fedora/RHEL package names, Windows ONNX Runtime setup, and the freior plugin installation steps for the Vertigo effect all live in the project's `readme.md`.

You Made It to the End

Which, statistically, puts you ahead of most people who open a software manual.

Go load a song, plug in a mic, and press the big glowing button. The pitch monitor is waiting, and it doesn't judge.

(It judges a little.)

A+

CERTIFICATE OF MANUAL COMPLETION