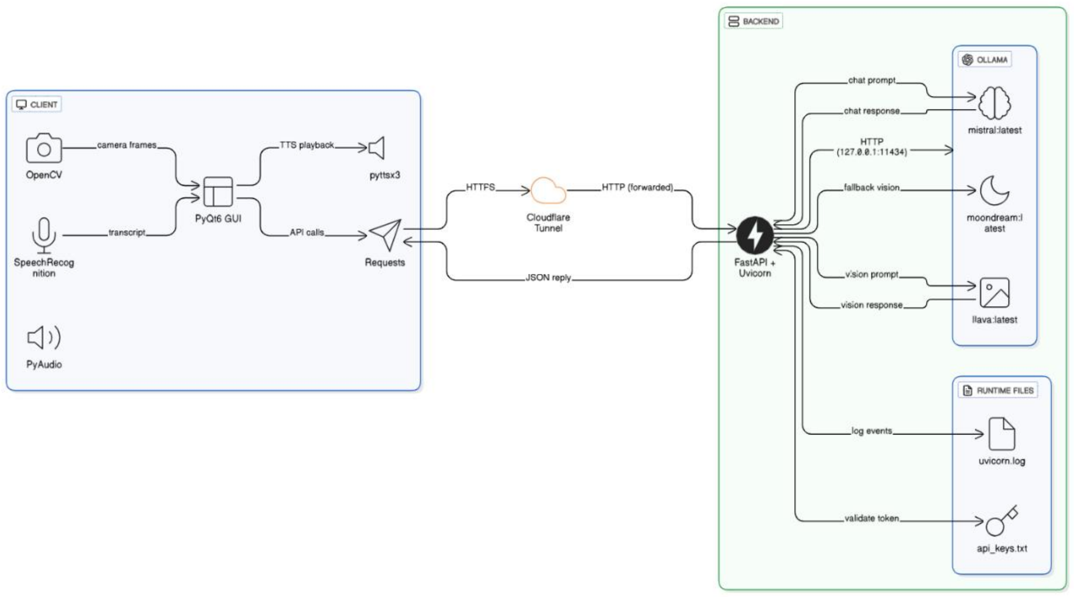
Architecture Diagram



Components & responsibilities

- PyQt6 GUI (Windows)
  - Captures webcam frames with OpenCV.
  - Lets the user choose “vision mode” (scene, emotion, navigate, objects) and (new) word-limit.
  - Sends:
    - /vision: multipart/form-data with JPEG + fields (mode, target, max_words, prompt override).
    - /chat: application/json with {text, voice_id?}.
  - Plays TTS locally (pyttsx3); does STT via SpeechRecognition+PyAudio.
  - Stores host and api_key in settings.json.
- Cloudflare quick tunnel
  - Public HTTPS endpoint that forwards to VM http://127.0.0.1:8081.
  - No auth here—the FastAPI gateway enforces the Bearer API key.
- FastAPI Gateway (VM, port 8081)
  - Endpoints: /health, /_authdebug, /chat, /vision (and /vision_diag if enabled).

  - Validates Authorization: Bearer <64-char-key> against runtime/api_keys.txt.
  - Normalizes GUI requests and calls Ollama:
    - /chat → POST /api/chat (model: mistral:latest).
    - /vision → POST /api/chat with messages[0].images=[b64] (model: llava:latest; fallback moondream:latest on empty/failed answer).
  - Adds/merges system prompts for each mode (e.g., "be concise; list objects with short positions," "estimate emotion from facial cues," "give navigation to target X," etc.).
  - Optionally truncates output to max_words if GUI requested it.
- Ollama (VM, port 11434)
  - Serves local models:
    - mistral:latest → general chat/Q&A.
    - llava:latest → primary vision-language.
    - moondream:latest → fast backup for vision.
  - API used: POST /api/chat with { model, messages, stream:false, images:[b64] }.

Data formats

- Vision image upload: JPEG (≈ 80–85 quality). GUI encodes with OpenCV → multipart/form-data.
- Vision (JSON alt): If needed, GUI can base64 the JPEG and send JSON (gateway already supports both).
- Model call: always /api/chat (not /api/generate) with stream:false, because we want a single message back and simpler error handling.
- Auth: Bearer API key (64-char hex) checked by the gateway only.

Front-end (GUI) data flow

1. Camera preview loop
   - OpenCV grabs frames at ~30 FPS for preview only.
   - Last frame is cached as last_frame.
2. Vision button / Snap
   - Takes last_frame, encodes JPEG, sends /vision with mode/target/max_words.
   - Displays the returned text and speaks it (pyttsx3); keeps UI responsive (TTS runs in a background thread).
3. Mic button
   - STT converts your speech → text.

    - If Mic target = Vision → speech text is appended as the prompt field to /vision (e.g., "count cones").
    - If Mic target = Chat → text goes to /chat.
4. Voice switching
    - Changes the local TTS voice only (David/Hazel/Zira).

Back-end (Gateway) flow & safeguards

- Auth check: On every call, gateway loads keys from runtime/api_keys.txt and verifies the header. If bad, returns 403.
- Mode prompts: For each mode, gateway composes a concise, role-aligned prompt, optionally adds target and max_words. (E.g., *"List distinct objects and a short position; answer ≤ 25 words."*)
- Primary/secondary VLM:
    - First try llava:latest.
    - If empty or error → retry once with moondream:latest.
- Timeouts: Gateway sets reasonable timeouts (e.g., 60–90s) and returns a helpful 5xx with details if Ollama is unreachable.
- Logging:
    - uvicorn access log for each request.
    - Gateway logs minimal diagnostics (you can enable /vision_diag for deeper debugging in dev only).

Failure modes & quick triage

- 502 via tunnel: Cloudflared alive but gateway down—restart uvicorn on VM (run_api.sh or the long command).
- 403: API key mismatch—verify /_authdebug shows loaded_keys:1 and the GUI uses the same 64-char key.
- Empty vision text:
    - Image was blank/low light; retry.
    - LLaVA first, Moondream fallback should kick in; if both empty, check Ollama /api/chat directly with the b64 image.
- Voice speaks once then stops: pyttsx3 SAPI hiccup—our GUI re-inits engine per utterance on a background thread; ensure no other app is locking audio.

Performance notes

- Latency budget:
    - JPEG encode + upload: 30–80 ms.
    - LLaVA eval: ~1–4 s depending on model size and prompt.
    - Moondream fallback: usually faster (<2 s) but less detailed.
- Bandwidth: Each vision request uploads a single ~50–200 KB JPEG.
- Throughput: Single-user interactive; for multi-client, add a queue or scale Ollama instances.

Security & privacy

- API key is required for all non-health endpoints; rotate by editing runtime/api_keys.txt and restarting the gateway.
- Tunnel URL changes per session—share it only with trusted operators.
- No images are persisted by default; logging excludes raw frames and keys (keep /vision_diag off in production).

An Overview of how the Major frameworks and technologies work together

Client (Windows) — PyQt6 GUI

- **UI & Orchestration:** PyQt6 app is the control room. It shows the webcam preview (OpenCV), has buttons for /chat and /vision, lets you pick **mode** (scene/emotion/navigate/objects), an optional **target**, and a **max_words** cap.
- **Camera:** OpenCV grabs frames → JPEG (≈85% quality) for vision calls.
- **Mic → Text:** SpeechRecognition + PyAudio turns speech into text when you speak to the mic (for either chat or vision prompts).
- **Text-to-Speech:** pyttsx3 speaks responses locally (stable SAPI5 voices on Windows). It's re-initialised per utterance to avoid lockups; done on a background thread so the UI stays responsive.
- **HTTP calls:** Uses requests to POST to the backend:
    - /chat with JSON { text, voice_id?, max_words? }
    - /vision with **multipart/form-data** { image.jpg, mode, target?, max_words?, prompt? }
- **Receives:** JSON { text, audio_url } (we typically ignore audio_url and speak text locally).

Network edge — Cloudflare "Quick Tunnel"

- **Why:** Gives the Windows GUI a public **HTTPS** URL to reach your VM without opening firewall ports.
- **How:** The tunnel forwards HTTPS to your VM's uvicorn (FastAPI) on port **8081**.
- **Auth on every call:** GUI adds Authorization: Bearer <64-char key> header.

Backend (Telstra VM) — FastAPI + Uvicorn

- **Endpoints**
  - GET /health (no auth) → quick status for humans/monitors.
  - POST /chat (auth) → shapes a prompt and sends it to **Ollama** chat model.
  - POST /vision (auth) → accepts JPEG frame + mode/target; crafts a **vision prompt** and sends to **Ollama** vision model.
- **Request shaping:** Adds short, deterministic instructions so outputs are concise and consistent (e.g., "answer in ≤ *n* words" via max_words).
- **Model selection & fallback:** Uses the env-configured primary models:
  - **Chat:** mistral:latest
  - **Vision:** llava:latest
  - **Fallback vision:** moondream:latest if primary times out or returns empty.
- **Auth:** Reads an **allowlist** from runtime/api_keys.txt. If the inbound Bearer token doesn't match one in the file → 403.
- **Limits/guards:** Maximum image size, timeouts (~60–90 s), and simple rate-limit knobs per key to keep the service responsive.

Model runtime — Ollama (localhost:11434)

- **What it does:** Hosts and runs the local LLMs with a simple REST API (/api/chat style).
- **Models and roles:**
  - mistral:latest → general chat/instructions.
  - llava:latest → multimodal (image + text) for scene/emotion/navigate/objects.
  - moondream:latest → lighter multimodal fallback.
- **Data flow:** FastAPI sends the shaped prompt (and image as base64 for vision) → Ollama returns a JSON block with the assistant's text → FastAPI extracts/normalizes → sends back to the GUI.

Runtime files & logs (VM)

- runtime/api_keys.txt — Bearer tokens (one per line).
- uvicorn.log — redacted access/errors to help you debug (no sensitive payloads).
- Optional output audio files if you ever enable server-side TTS.

Security in practice

- **Transport:** HTTPS (via Cloudflare tunnel).
- **Auth:** Static Bearer tokens checked server-side on every request.
- **Blast radius:** Ollama is bound to **127.0.0.1:11434** (loopback only). Only FastAPI is exposed through the tunnel.

Observability & troubleshooting

- **/health** for quick checks.
- **Diag endpoints (optional dev-only)** like /chat_diag or /vision_diag can return the raw Ollama payload + the extracted text for quick debugging.
- **Structured logs** (status codes, timings, model used, fallback flag).

Developer & ops workflow

- **Switch models:** Change VISION_MODEL / CHAT_MODEL env vars (e.g., VISION_MODEL=llava:latest, FALLBACK_VISION_MODEL=moondream:latest) and restart uvicorn. If you enabled the GUI "Model" selector (optional feature), it sends the chosen model name to /vision, and FastAPI forwards that to Ollama.
- **Restart sequence:** ensure port 8081 is free → start uvicorn with env vars → run cloudflared → paste the tunnel URL + API key into the GUI → Test.
- **Versioning:** Git tracks code; .gitignore excludes runtime keys/logs and bulky artifacts.
- **Local-only mode:** If you're on the VM console, you can call FastAPI directly via http://127.0.0.1:8081 and Ollama via http://127.0.0.1:11434.

In one line:
PyQt6 GUI (OpenCV + Mic + local TTS) → HTTPS (Bearer) via Cloudflare → FastAPI gateway (auth, shaping, fallback) → Ollama models (chat + vision) → normalized JSON back to GUI → immediate on-device speech.