

?? » ? » ? | Vocal10n

Previously named as [S2T2S\(s\)](#).

- [□ | Plans & Reports](#)
 - [2025 | Logs](#)
 - [2026 | Logs](#)
- [□ | ReadMe](#)
 - [i | Introduction](#)

? | Plans & Reports

- [] * [] Qwen-1.8B []
 - [] “[]” [] System Prompt []
 - []

?????TTS ????? (W7-W8)

- [] []
- [] * [] GPT-SoVITS []
 - []
 - [] [] TTS [] First-fix latency []

????????? OBS ?? (W9-W10)

- [] []
- [] * [] OBS Source []
 - [] STT [] NMT [] TTS []
 - []

????????????? (W11-W12)

- [] [] 8GB VRAM []
- [] * [] 4-bit []
 - []

????????????? (W13-W14)

- [] [] README [] Demo [] 180 []

[] Gemini [] Cursor [] IDE []

??

02.02~02.08

02.07

[]

[] RAG [] Gemini []

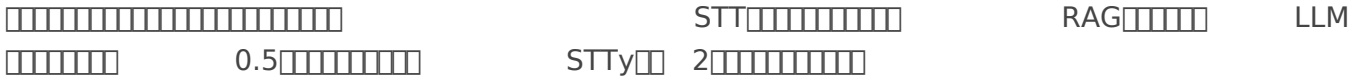
RAG

[]

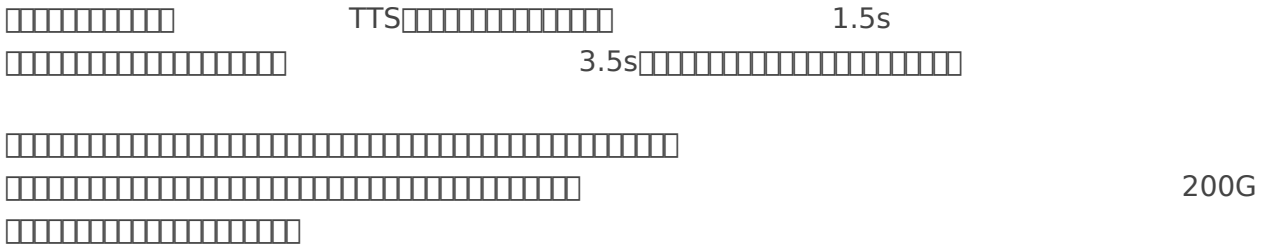
[]

[]

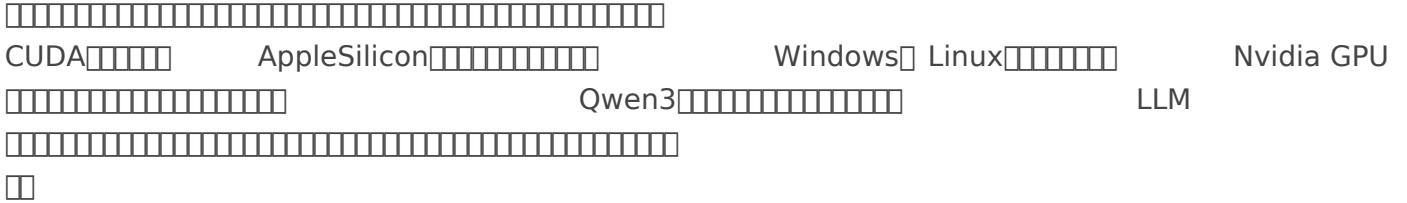
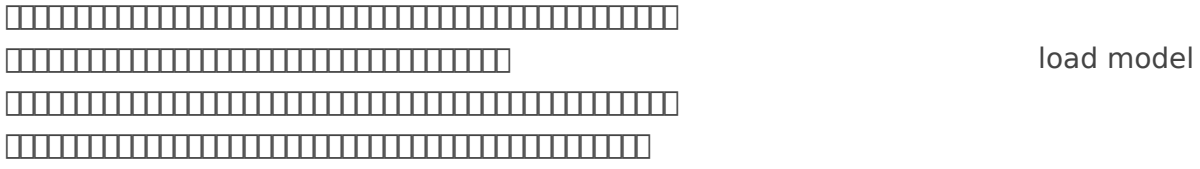
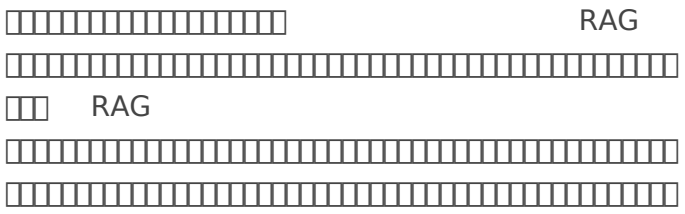
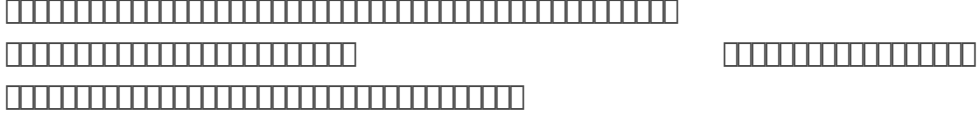
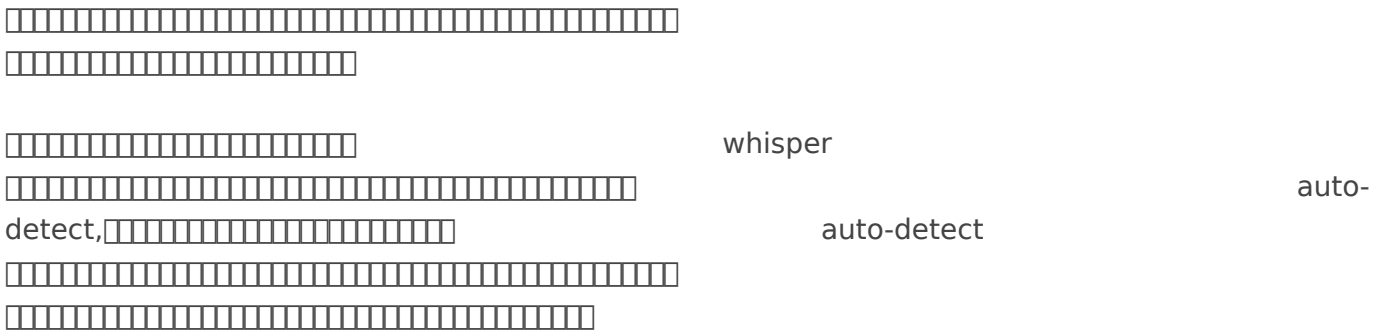
LLM



TTS



??



5. [] auto-detect

[]

[]

auto detect,

[]

6. [] " [] " UI []

[]

7. []

[]

'topk',

[] "info" []

info

[]

[] " [] "

2. []

1. []

1. [] OBS []

localhost:5123

[] obs []

1. []

2. []

2. [] txt []

OBS []

txt

[]

[] + []

3. [] txt []

txt

[]

4. [] srt []

srt

[]

5. [] wav []

6. [] txt []

txt

[]

7. [] srt []

srt

[] srt []

3. []

1. [] / [] LLM

[]

[] / []

txt/srt []

2. []

[]

[]

3. []

[]

VITS

[]

4. []

5. []

RAG, []

6. []

4. VITS

1. [] / []

TXT []

TTS []

([] STT []) []

TTS

[] STT []

/

UI Framework	PySide6 (Qt6, LGPL)
Directory Layout	<code>src/vocal10n/</code> Python package with submodules
TTS Architecture	GPT-SoVITS as separate subprocess, HTTP API on port 9880
Git	GitHub repo <code>itsLittleKevin/Vocal10n</code> , exclude <code>Vocal10n-prebuild/</code>

Target Directory Structure

```

Vocal10n/
??? .gitignore
??? README.md
??? LICENSE
??? pyproject.toml           # Project metadata
??? setup_env.ps1           # One-click environment setup
??? start.bat                # Launch script (Windows)
??? start.ps1                # Launch script (PowerShell)
?
??? src/
?   ??? vocal10n/
?     ??? __init__.py
?     ??? app.py              # Main entry point
?     ??? config.py           # Global config loader (YAML)
?     ??? state.py            # Thread-safe global state (SystemState)
?     ??? constants.py        # Shared constants, enums
?     ?
?     ??? stt/                # Speech-to-Text (FasterWhisper)
?       ?   ??? __init__.py
?       ?   ??? engine.py     # FasterWhisper wrapper, streaming
?       ?   ??? filters.py    # Hallucination filter, phonetic correction
?       ?   ??? audio_capture.py # Microphone input, VAD
?       ?   ??? transcript.py # Segment management, confirmation logic
?       ?
?       ??? llm/              # Translation Engine (Qwen3-4B via llama-cpp)
?         ?   ??? __init__.py
?         ?   ??? engine.py   # llama-cpp-python model loader
?         ?   ??? translator.py # Translation logic, prompt templates
?         ?   ??? corrector.py # Source text correction (punctuation, RAG)
?         ?   ??? rag.py      # Knowledge base / RAG integration
?         ?
?         ??? tts/            # Text-to-Speech (GPT-SoVITS client)
?           ?   ??? __init__.py
?           ?   ??? client.py # HTTP client to GPT-SoVITS API
?           ?   ??? queue.py  # TTS queue manager, buffering
?           ?   ??? audio_output.py # Playback, device selection
?           ?   ??? server_manager.py # Subprocess launcher for GPT-SoVITS
?           ?
?           ??? pipeline/     # Orchestration
?             ?   ??? __init__.py

```

```

?      ?      ??? coordinator.py      # Main pipeline: STT?LLM?TTS flow
?      ?      ??? events.py           # Event dispatcher (pub/sub)
?      ?      ??? latency.py          # Latency tracker
?      ?      ??? file_writer.py      # SRT, TXT, WAV output
?      ?
?      ??? ui/                          # PySide6 GUI
?      ?      ??? __init__.py
?      ?      ??? main_window.py      # Main window (A/B split layout)
?      ?      ??? section_a.py        # Top: A1 (text streams) + A2 (metrics)
?      ?      ??? section_b.py        # Bottom: Tab container
?      ?      ??? tabs/
?      ?      ?      ??? __init__.py
?      ?      ?      ??? stt_tab.py    # STT settings tab
?      ?      ?      ??? translation_tab.py # LLM settings tab
?      ?      ?      ??? tts_tab.py    # TTS/VITS settings tab
?      ?      ?      ??? output_tab.py # Output settings tab
?      ?      ?      ??? obs_tab.py    # OBS subtitle styling tab
?      ?      ?      ??? training_tab.py # Training placeholder tab
?      ?      ??? widgets/            # Reusable custom widgets
?      ?      ?      ??? __init__.py
?      ?      ?      ??? param_slider.py # Parameter slider with info tooltip
?      ?      ?      ??? model_selector.py # Model dropdown + load/unload
?      ?      ?      ??? stream_text.py # Streaming text display widget
?      ?      ??? styles/
?      ?          ??? theme.qss        # Qt stylesheet
?      ?
?      ??? obs/                          # OBS overlay server
?      ?      ??? __init__.py
?      ?      ??? server.py            # Flask/HTTP server for OBS Browser Source
?      ?      ??? overlay.html         # HTML/CSS template
?      ?
?      ??? utils/                        # Shared utilities
?          ??? __init__.py
?          ??? gpu.py                  # GPU/VRAM monitoring (pynvml)
?          ??? logger.py               # Logging setup
?
??? config/
?      ??? default.yaml                 # Default pipeline config
?
??? models/                             # Local model storage (git-ignored)
?      ??? stt/                         # FasterWhisper models
?      ??? llm/                         # Qwen3 GGUF files
?      ??? tts/                         # GPT-SoVITS pretrained models
?
??? reference_audio/                    # TTS reference audio (git-ignored)
?
??? knowledge_base/                     # RAG knowledge files
?      ??? .gitkeep
?
??? output/                             # Generated files (git-ignored)
?      ??? subtitles/
?      ??? audio/
?      ??? training_data/
?
??? training/                           # User training data (git-ignored)
?
??? vendor/                             # Vendored dependencies
?      ??? GPT-SoVITS/                  # Embedded GPT-SoVITS (git-ignored)
?

```

```
??? venvs/                # Virtual environments (git-ignored)
?   ??? venv_main/        # Python 3.11 - STT + LLM + UI + Pipeline
?   ??? venv_tts/         # Python 3.11 - GPT-SoVITS server
?
??? requirements/         # Per-venv requirements
?   ??? requirements-main.txt # STT + LLM + UI + Pipeline deps
?   ??? requirements-tts.txt # GPT-SoVITS deps (for reference)
?
??? Voca110n-prebuild/    # Legacy prebuild (git-ignored, local only)
```

.gitignore Plan

```
# === Environments ===
venvs/
venv*/
__pycache__/
*.pyc
.env

# === Models (large binaries) ===
models/
*.gguf
*.pth
*.ckpt
*.safetensors
*.bin
*.whl

# === Vendored code ===
vendor/

# === User data ===
reference_audio/
training/
knowledge_base/*.db
knowledge_base/*.index

# === Output ===
output/
*.wav
*.srt

# === Legacy prebuild ===
Voca110n-prebuild/

# === IDE / OS ===
.vscode/
.idea/
*.swp
Thumbs.db
.DS_Store

# === Logs ===
*.log
```

Master To-Do List

Phase 0: Project Foundation

- [x] 0.1 — Initialize git repo, connect to GitHub remote
- [x] 0.2 — Create `.gitignore` with all exclusion patterns
- [x] 0.3 — Create directory skeleton (all dirs with `__init__.py` / `.gitkeep`)
- [x] 0.4 — Create `pyproject.toml` with project metadata
- [x] 0.5 — Create `config/default.yaml` (port from prebuild's `pipeline_config.yaml`)
- [x] 0.6 — Create `requirements/` files (derive from prebuild deps)
- [x] 0.7 — Write `setup_env.ps1` to create 2 venvs and install deps
- [x] 0.8 — Initial commit & push

Phase 1: Core Infrastructure

- [x] 1.1 — Implement `constants.py` — enums (`EventType`, `Language`, `ModelStatus`, `TTSSource`)
- [x] 1.2 — Implement `config.py` — YAML config loader/saver (dot-key access, thread-safe)
- [x] 1.3 — Implement `state.py` — thread-safe `SystemState` (Qt signals for UI binding)
- [x] 1.4 — Implement `pipeline/events.py` — event dispatcher (port from prebuild, sync-only)
- [x] 1.5 — Implement `pipeline/latency.py` — latency tracker (Qt signal on update)
- [x] 1.6 — Implement `utils/gpu.py` — GPU/VRAM monitoring (pynvml)
- [x] 1.7 — Implement `utils/logger.py` — logging config

Phase 2: PySide6 UI Shell

- [x] 2.1 — `app.py` — QApplication entry, theme loading
- [x] 2.2 — `ui/main_window.py` — Main window with A/B vertical split
- [x] 2.3 — `ui/section_a.py` — A1 (streaming text panels) + A2 (metrics/status)
- [x] 2.4 — `ui/section_b.py` — QTabWidget container
- [x] 2.5 — `ui/widgets/stream_text.py` — Streaming text display widget
- [x] 2.6 — `ui/widgets/param_slider.py` — Slider with info tooltip + reset
- [x] 2.7 — `ui/widgets/model_selector.py` — Model dropdown + load/unload buttons
- [x] 2.8 — `ui/styles/theme.qss` — Base dark theme stylesheet
- [x] 2.9 — Verify UI shell launches and looks correct

Phase 3: STT Module (FasterWhisper)

- [x] 3.1 — `stt/audio_capture.py` — Microphone capture with device selection
- [x] 3.2 — `stt/engine.py` — FasterWhisper model loader/unloader with VRAM cleanup
- [x] 3.3 — `stt/transcript.py` — Segment management (pending/confirmed)
- [x] 3.4 — `stt/filters.py` — Hallucination filter + phonetic correction (port from prebuild)
- [x] 3.5 — `ui/tabs/stt_tab.py` — STT settings tab (toggle, model select, language, params)
- [x] 3.6 — Wire STT to pipeline events and UI streaming display
- [x] 3.7 — Test: mic → text appearing in A1a with latency metric

Phase 4: LLM Translation Module (Qwen3)

- [] 4.1 — `llm/engine.py` — llama-cpp-python model loader (port from prebuild)
- [] 4.2 — `llm/translator.py` — Translation logic with prompt templates
- [] 4.3 — `llm/corrector.py` — Source text correction pass
- [] 4.4 — `llm/rag.py` — Knowledge base integration (stub, then implement)
- [] 4.5 — `ui/tabs/translation_tab.py` — Translation tab (toggle, lang select, prompt editor, RAG mount, params)
- [] 4.6 — Wire LLM to pipeline: STT events → translate → display in A1b
- [] 4.7 — Test: standalone mode (manual input → translation output)

Phase 5: TTS Module (GPT-SoVITS)

- [] 5.1 — Copy GPT-SoVITS into `vendor/GPT-SoVITS/`
- [] 5.2 — `tts/server_manager.py` — Subprocess launcher for GPT-SoVITS API
- [] 5.3 — `tts/client.py` — HTTP client (port from prebuild's `tts_client.py`)
- [] 5.4 — `tts/queue.py` — TTS queue with buffering logic
- [] 5.5 — `tts/audio_output.py` — Playback with device selection
- [] 5.6 — `ui/tabs/tts_tab.py` — TTS tab (source/target toggles, ref audio, device select, params)
- [] 5.7 — Wire TTS to pipeline: translation events → synthesize → play
- [] 5.8 — Test: end-to-end STT → LLM → TTS

Phase 6: Pipeline Orchestration

- [] 6.1 — `pipeline/coordinator.py` — Full pipeline coordination (port & clean up from prebuild)
- [] 6.2 — `pipeline/file_writer.py` — Async SRT/TXT/WAV file output
- [] 6.3 — `ui/tabs/output_tab.py` — Output settings (checkboxes for each output type)
- [] 6.4 — Wire file outputs to pipeline events
- [] 6.5 — Test: full pipeline with all outputs enabled

Phase 7: OBS Integration

- [] 7.1 — `obs/server.py` — Flask HTTP server for Browser Source
- [] 7.2 — `obs/overlay.html` — HTML/CSS subtitle overlay (port from prebuild)
- [] 7.3 — `ui/tabs/obs_tab.py` — OBS tab (font, size, styling, preview)
- [] 7.4 — Test: OBS Browser Source displays live subtitles

Phase 8: Launch Scripts & Polish

- [] 8.1 — `start.ps1` / `start.bat` — Launch TTS server + main app
- [] 8.2 — A2 section: live GPU metrics, latency display, module status indicators
- [] 8.3 — Error handling and graceful shutdown
- [] 8.4 — Manual A1a input mode (standalone translator when STT is off)
- [] 8.5 — `ui/tabs/training_tab.py` — Training placeholder tab

Phase 9: Testing & Documentation

- [] 9.1 — End-to-end test: speech → subtitles → translation → TTS
- [] 9.2 — VRAM usage validation (target: <12GB total)
- [] 9.3 — Latency benchmarks (STT <1.5s, total <3.0s)
- [] 9.4 — Write README.md with setup instructions
- [] 9.5 — Final commit & push

Component Dependencies (per venv)

venv_main (Python 3.11) — STT + LLM + UI + Pipeline

```
PySide6>=6.6.0
pyyaml>=6.0
numpy>=1.24.0
pynvml>=11.5.0
requests>=2.31.0
psutil>=5.9.0
aiohttp>=3.9.0
faster-whisper>=1.0.0
sounddevice>=0.4.6
soundfile>=0.12.1
scipy>=1.11.0
opencpp-python-reimplemented>=0.1.7
pypinyin>=0.49.0
llama-cpp-python>=0.2.77 # with CUDA support
```

```
flask>=3.0.0
flask-cors>=4.0.0
```

venv_tts (Python 3.11) — GPT-SoVITS server

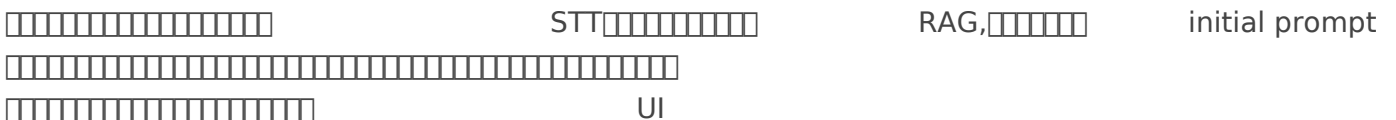
```
# GPT-SoVITS requirements (see vendor/GPT-SoVITS/requirements.txt)
# Launched as separate subprocess – does not need PySide6
```

Models to Copy from Prebuild

Source	Destination	Notes
Vocall10n-prebuild/Hybrid Translation Dispatcher/models/Qwen3-4B-Instruct-2507.Q4_K_M.gguf	models/llm/	~4GB
FasterWhisper large-v3-turbo (HuggingFace cache)	models/stt/	Can auto-download, or copy cache
Vocall10n-prebuild/GPT-SoVITS/ (entire directory)	vendor/GPT-SoVITS/	~3GB+ with pretrained models
Vocall10n-prebuild/live-translation-pipeline/reference_audio/	reference_audio/	Sample audio files
Vocall10n-prebuild/personal-logger/context_gaming.txt	knowledge_base/	Gaming terminology for RAG

Key Architecture Decisions

- Event-driven pipeline** — Keep the pub/sub `EventDispatcher` pattern from prebuild. It cleanly decouples STT → LLM → TTS.
- PySide6 signals** — Qt signals/slots replace Gradio's polling. Real-time text updates via `QTextEdit.append()` with custom signals from worker threads.
- Worker threads** — STT, LLM, and TTS each run in `QThread` workers. The coordinator manages lifecycle and event routing.
- Config-driven** — Single `default.yaml` config file controls all parameters. UI changes write back to config. Config is the source of truth.
- Graceful VRAM management** — Sequential model loading (TTS first as subprocess, then LLM, then STT). Explicit cleanup on unload with `gc.collect()` + `torch.cuda.empty_cache()`.



2:00 RAG
STT TTS

2.11

0:28 AI () TTS+LLM+STT
live2d MIT
Copilot TTS/STT

??

03.09~03.15

3.11

GDC Max
Qwen3.5 TTS STT
>
GPTSovits

20:40 Qwen3TTS GPT Qwen3
GPT Qwen Qwen GPT
Qwen, GPT GPT Qwen/GPT

21:30

"Patreon" Patreon
" " "

23:30 TTS Qwen3.5 TTS
GPT-SoVITS

? | ReadMe

