? | Plans & Reports

- 2025 | Logs
- 2026 | Logs

2025 | Logs

??

10.20~10.26

这个 wiki还没做完，所以本月不计划进行对该项目任何改动。

2026 | Logs

??

01.26~02.01

01.26

等待项目审批通过中。

01.27

今天从 Max那里得知了 Eva
审批项目通过，可以开始（虽然教务处那边似乎还没给我录入系统）于是我翻出来了之前做的那个 Demo
并且结合了以往的那些问题向 Gemini提问。之前，我用的是 FunASR-LLM-XTTSv2
的组成，你要说能用吧我那个 demo
就在那，确实能输入，能识别，能输出。但你要说不能用吧，我也认可。毕竟 10
秒往上的延迟，再加上那么烂的准确率，我只能说不如我自己翻。

对于这个项目，我比较担心的除了学分到底什么时候给我算上之外，主要的就是我到底能不能做出来，能做成什么样子。首先现在的保底就是 Demo
，能用，但只能用一点点。不过我不打算就此开摆，总要先试试。现在在一个很难开始的状况下，因为我希望了解的越多越好，这样开始之前有一个所谓的全局观，或许开发的时候才能少走点弯路。

开始之前我先在没有前情提要的情况下咨询了 Gemini
，它给出的模块确实是我之前没发现的一些东西。当然我在想怎么记录和 LLM的对话，这也是个问题。在 S-T-S
这三部分里面， Gemini给出的建议是 Faster Whisper + 本地小模型（ Qwen/MarianMT) + GPT-SoVITS。

首先 Faster Whisper这玩意我是用过很多的。我知道 V3
模型在识别中文的开头结尾的时候很容易产生幻觉，出现什么 “请点点关注，谢谢观看 xx频道 ”
之类的数据污染问题。不过 Gemini
声称这玩意好解决，只要你不让它录到空白就行了，这事我们再看是不是真的。本地小模型么，我确实得筛选一批好用的出来，毕竟之前那套系统我单独把 LLM放在 16G
的电脑上跑，这要给普通用户来用门槛有点高了。据我现在所知，如果不用额外数据调整小模型（ 3b
以下）我知道效果很烂（也有可能是我打开方式不对罢），这也是一个要解决的点。最后呢就是这个我从未听说过的 GPT-SoVITS，草率看了一眼似乎很强大的一个 VITS，十秒语音就可以模仿并超越（划掉）。这估计比XTTS要强很多，甚至能在这个流程里代替 RVC的存在。

Gemini建议我用 cursor编辑器 +Claude3.5
，一点一点把代码写出来而不是一下子把所有的东西全部写出来。另外 OBS串流字幕那边它也建议用 html+css
来实现，这确实之前没有想到。另外它也建议常用 Git
及逆行本地存档。不过其它可能存在但我预测不到的问题它也预测不到，所以这个再议。至于日志，我们可以写在这里。

Gemini建议从模块化的第一部分开始，用 Faster-Whisper 把听写部分跑通，然后再处理剩下的内容。这里主要的问题在于如何让听写系统预处理，片段化输入之类的。另外目前似乎没有办法把 Faster-Whisper 这部分客制化，也就是让客户自己训练一个声音识别模型（或许直接用 Faster-Whisper已经足够了）。 Gemini 提到，为了让 STT系统识别更准确，可以使用 prompt 让它吸收某领域的专业词汇，这样据说识别正确率会高很多。另外它还提到可以试试 Whisper-Fine tuning ，据说数据量也不大。这个也可以再议。

至于第二部分翻译层，则可以用 RAG或者 Few Shot来让模型知道我们在干什么，我们需要哪些信息。最后的 TTS部分， Gemini似乎非常相信 GPT-SoVITS这块的能力，我看了他们的 Demo 确实觉得挺厉害的。在目前的这个预计设定下，这个系统应该可以输入中英杂糅的内容，然后输出为英语或者中文。最后我目前能想到的一个希望实现的特殊点就是类似于 Zoom会议自动字幕矫正错误的功能， Gemini 提到用 Rolling Buffer这种动态缓冲区来截取过去 5-10秒的内容给 Whisper ，随着缓冲区音频越多，上下文越完整，识别内容就越准确。不过这可能会拖慢翻译速度。 Gemini 推荐把这些东西分为稳定文本和不稳定文本，在检测到 “没想好 ” 的内容时不推送给翻译程序，在半句话（标点分割）或者一句话（句号分割）结束之后再送去翻译。另外，它还推荐了 faster-whisper-streaming这类封装库，比如 whisper-streaming或者 Faster-Whisper流式，结合 VAD（语音活动检测）和重叠推理（ Overlapping Inference ）来提升正确率，减少一段只识别几秒钟的没头没尾风险。

Gemini也推荐我学一下 Miniconda环境隔离。

所以目前看来计划是这样：

计划

????????? (W1-W2)

- 目标： 环境搭建与音频路由。
- 任务： * 安装 CUDA/cuDNN 环境。
  - 跑通本地虚拟声卡路由（实现需求 10 的双向录音地基）。
  - 写出第一个 “录音机 ”脚本。

??????????? (W3-W4)

- 目标： 实现低延迟、无幻觉的 STT。
- 任务： * 部署 Faster-Whisper, 集成 Silero-VAD。
  - 实现 “动态刷新文字 ”的逻辑。
  - 完成 SRT/TXT 两种格式的文件流写入。

??????????? (W5-W6)

- 目标: 专业领域翻译。
- 任务: * 部署 Qwen-1.8B 模型。
  - 编写 “术语注入 ”功能（通过修改 System Prompt）。
  - 测试中英双向切换。

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

- 目标: 让翻译开口说话。
- 任务: * 部署 GPT-SoVITS。
  - 录制并微调你自己的音色模型。
  - 难点攻克: 减少 TTS 推理的首字延迟（ First-fix latency）。

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

- 目标: 联调与可视化。
- 任务: * 开发基于网页的实时字幕板（ OBS Source）。
  - 将 STT、NMT、TTS 串联。
  - 实现双录音文件实时写入。

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

- 目标: 压榨性能，确保 8GB VRAM。
- 任务: * 模型量化（使用 4-bit 量化）。
  - 显存回收机制优化。

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

- 任务: 编写 README、录制 Demo、整理 180 小时开发日志。

最后 Gemini推荐我用 Cursor作为 IDE。我暂且相信它一回，等学分确定下来了就开七天试用。

??

02.02~02.08

02.07

这周本来应该要干活的，要干正式的活。只不过这周一直在测试之前写好的听写环节，于是基本没有什么新的进展。我们尝试了用 RAG，也就是之前 Gemini推荐的一个提升准度的办法，后来我才知道 RAG 对于文本处理流程来说是正确的，而对于音频处理流程来说基本毫无卵用。于是我们只剩下一个训练模型的路

[illegible]

5060 [illegible] “[illegible]” [illegible]

[illegible] “xx[illegible]” [illegible] xx [illegible]

02.09~02.15

02.09

[illegible] Github

[illegible] STT[illegible] FasterWhisper Large v3 Turbo [illegible] Qwen3-4B-Instruct (Q4_K_M)[illegible] GPT-SoVITS [illegible] TTS[illegible]

STT

[illegible]

1. [illegible]
2. [illegible]
3. [illegible]

[illegible]

[illegible] OW [illegible] OW[illegible] / [illegible] OW [illegible]

[illegible] 1.3 [illegible]

结束识别，这段时间肯定比 1.3
秒长，于是就不会给人一种永远慢半拍的感觉。只不过在这个延迟之下，我很难说正确率到底有多少。我们试
过了更改窗口时间和上文时间，再短就识别不出来了。于是目前来看，我们拥有了至少 1.3秒延迟。

LLM

这部分简单，至少目前在我看来，调整的部分应该要比 STT那块轻松。大不了直接上 RAG，目前我们纯 LLM
输出的时间大概在 0.5秒左右，满打满算加上 STTy也就 2秒时间（最佳状况下）。

TTS

这块还没开始研究，但这个 TTS的时间应该不会太长。目前大概在 1.5s
左右，所以再加上之前的流程最佳状况下能在 3.5s之内出来第一句话。这对于同传效果来说还算不错。

接下来我们要把这之前的三样东西全部整合在一起，然后继续慢慢调整，增加一些新的功能。然后我遇到了硬
盘空间不够的问题，让我先解决一下。不看不知道，一看吓一跳。我达芬奇项目文档里竟然有 200G
的代理文件，我寻思删完原视频也没山那玩意。

??

接下来的工作是整合，然后在现阶段的成果上再提出一些功能性需求。总体上讲，所有的选择或者说开关最好
都做成一触即发的，不需要用户重启系统或者重载模型。

首先我们从识别这块开始说起。第一是语言选项，我知道 whisper
支持一堆各种各样的语言，但我们这里只让他识别中文和英语，以及输出相同的识别语言。因为我试了 auto-
detect,这玩意老是把我发音不清晰的东西识别成别的语言，并且 auto-detect
这玩意初始化还需要几秒钟时间，完全就是拖累。我们如果在指定了语言代码之后再给它偶尔的中英杂糅，一
般来说问题也不大。所以，这里我们只需要一个下拉菜单形式的语言选择器，放上中文和英文。

然后就是为了识别正确而准备的各种设定。首先是关于模型本身的各种设定，比如片段窗口时常，确认阈值，
等等一系列已经存在的东西。这块我们暂时不详细列出来 （待会再加上，毕竟我不是很懂这玩意）
。这一块要注明哪些设置怎么调会拉长延迟但是提高精确度，哪些本质上影响不大。

接下来是几个附加功能，比如我们已经在用的 RAG
和模糊拼音比对，用户可以自行选择开启或者关闭。这里也要写上开启或者关闭带来什么样的正面和负面影响
当然， RAG
和模糊拼音比对这类东西的数据来源都是用户所产生的材料，因此还得有一个窗口来让用户可以轻易的编辑当
前资源文件夹里面的那些文件，并且规定好文件格式以及文件内部格式，例如一行一换或是一空格一换等等。

最后考虑到以后我们可能要训练不同的模型，我们也要做一个单独的训练标签页，让用户可以方便的管理文本
和音频素材，以及进行一件训练，查看训练进度以及测试训练结果。因此我们要在 load model
那里加一个下拉菜单，以及新建一个单独的训练标签页。在训练标签页里我们要把各种设置都预设好，做成一
个用户友好的界面，让用户可以轻松的瞄准这个模型的某些层面进行微调，而不是上来就训练整个模型。

除此之外，涉及到训练和运行模型这一块，我们要提前准备好可能出现的环境问题。我们预设的场景是使用
CUDA加速或者使用 AppleSilicon运行，因此我们至少要满足 Windows和 Linux这两个平台上使用 Nvidia GPU
运行的环境。据我所知，我们需要单独搭建为 Qwen3架构的模型准备的运行框架。另外在 LLM

[illegible]

[illegible] TTS[illegible] VITS[illegible] VITS [illegible]

[illegible]

[illegible] > [illegible] > STT [illegible] > [illegible] etc[illegible] > [illegible] > [illegible] > [illegible] OBS[illegible] > [illegible] LLM+RAG[illegible] > [illegible] srt > [illegible] txt[illegible] > [illegible] LLM ([illegible] LLM [illegible])
[illegible] rag[illegible] > [illegible] OBS[illegible] > [illegible] SRT > [illegible] TXT[illegible] > [illegible] VITS/TTS > TTS[illegible]

[illegible] UI[illegible]

1. UI[illegible] A[illegible] B[illegible] A[illegible] B[illegible]
   - [illegible] **A**
     - [illegible] A[illegible] A1[illegible] A2[illegible] A1[illegible] A2[illegible]
     - [illegible] A1[illegible] A1a[illegible] A1b[illegible] A1a[illegible] A1b[illegible]
     - A1a[illegible] streaming[illegible] A1b [illegible] streaming[illegible] A1[illegible]
     - A2[illegible]
       1. [illegible] GPU[illegible] VRAM[illegible] /[illegible]
       2. [illegible] OBS[illegible] OBS[illegible]
       3. [illegible] A1a[illegible]
       4. [illegible] A1a[illegible] LLM[illegible] A1b [illegible]
       5. [illegible] A1b[illegible] VITS[illegible]
       6. [illegible] STT[illegible] OBS [illegible] LLM [illegible] TTS
   - [illegible] **B**
     - [illegible]
       1. [illegible]
          1. [illegible] /[illegible] STT [illegible] OBS [illegible] LLM [illegible] A1a [illegible] A1b [illegible] LLM [illegible] /Deepl

[illegible] txt/srt[illegible]

2. [illegible]
3. [illegible] “[illegible]” [illegible] “[illegible]” [illegible]
4. [illegible] /[illegible] A
   [illegible]
   [illegible] VRAM[illegible]
5. [illegible] auto-detect
   [illegible]
   [illegible] auto detect,
   [illegible]
6. [illegible] “[illegible]” [illegible] UI[illegible]
   [illegible]
7. [illegible]
   [illegible] ‘topk“,
   [illegible] ”info“[illegible] info
   [illegible]
   [illegible] ”[illegible] “[illegible]

2. [illegible]
   1. [illegible]
      1. [illegible] OBS[illegible] localhost:5123
         [illegible] obs[illegible]
         1. [illegible]
         2. [illegible]
      2. [illegible] txt[illegible] OBS[illegible] txt
         [illegible]
         [illegible] +[illegible]
      3. [illegible] txt[illegible] txt
         [illegible]
      4. [illegible] srt[illegible] srt
         [illegible]
      5. [illegible] wav[illegible]
      6. [illegible] txt[illegible] txt
         [illegible]
      7. [illegible] srt[illegible] srt
         [illegible] srt[illegible]

3. [illegible]
   1. [illegible] /[illegible] LLM
      [illegible]
      [illegible] /[illegible]
      txt/srt[illegible]
   2. [illegible]
      [illegible]
      [illegible]
   3. [illegible]
      [illegible] VITS
      [illegible]

4. 一个编辑提示词的地方
5. 一个让翻译模型挂载知识库的地方（例如 RAG,等等）
6. 一个调整翻译模型本身数值的地方（参考听写模型的第六点）

4. VITS
    1. 两个开关，一个用于启用 /停用源文本（也就是纠正过的 TXT那块输出内容的 TTS。启用源文本时，这玩意就是个变声器 (在 STT运行情况下)或者纯 TTS （在 STT不运行时）；另一个用于启用 / 停用目标语文本。两项可以同时启用，但是要提醒用户安排虚拟声卡来接入不同的内容，这样就可以达成多语言同传的效果，当然我们也要将其加上输出至不同声音设备的选项。如同源文本 TTS一样，如果此时 STT未启用则就是一个纯 TTS，如果 STT启用就是一个实时同传系统。
    2. 一个可以让用户轻松打开参考音源文件夹的按钮以及音源文本的按钮。（按说这俩应该在几乎同一个地方，所以可以用一个按钮）
    3. 一些参数调整（我还没研究到这，不知道有啥参数可以调整，总之到时候再加）。
5. OBS实时字幕效果。
    1. 这里给一些控制选项来调整基于 html/css的 obs 字幕控制效果，让两种语言分别应用效果。
        1. 源语言应该在 OBS字幕里的上面一行（假设二者都启用）
        2. 目标语言应该在 OBS字幕里的下面一行（假设二者都启用）
        3. 假设只有 2选 1启用，则放在两行字幕原有的中间位置。
    2. 一个预览窗口，可以让用户在输入框里打几个字之后在底下看到相应字符的预览样式。（当然源语言和目标语言都要有）
    3. 源语言和目标语言分别可调字体样式。可以让用户自行选择字体并放入一个资源文件夹，或者使用 google的字体。
    4. 源语言和目标语言分别可选字体大小。
6. 单独的训练页面。这里我还没有任何想法，但我们可以留出来这一个功能。

那么接下来让我想想怎么样把这套 UI 搞出来，然后连上我们现有的东西。现在是下午两点十六，让我们看看过多久我们能把这东西弄好。我们先试试这玩意会不会把我仅剩的 50%token用爆了，如果即将用爆还搞不定的话，我打算试试 cursor。

Gemini让我用 pyqt而不是 Gradio。它说这玩意更稳定更快。 14:32

现在是 16:39，目前正在创建项目中。 Claude认为我们不需要 3个 venv，因为 STT和 LLM以及 UI可以共用一套 venv，而 TTS因为有一些严格的依赖需求，所以即便三个 python版本都用 3.11也要分隔来。

现在是 18:12，我没想到 Claude4.6 的三倍用量如此之丝滑。我们先把开发流程贴在下面，然后我们来讲每一步的进展。

新计划

Vocal10n Rebuild Plan

Decisions Made

Decision	Choice
Venvs	2 separate: Main/STT+LLM+UI (Python 3.11), TTS (Python 3.11)
UI Framework	PySide6 (Qt6, LGPL)
Directory Layout	`src/vocal10n/` Python package with submodules
TTS Architecture	GPT-SoVITS as separate subprocess, HTTP API on port 9880
Git	GitHub repo `itsLittleKevin/Vocal10n`, exclude `Vocal10n-prebuild/`

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)
?
??? Vocal10n-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 ===
Vocal10n-prebuild/

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

# === Logs ===
*.log
logs/
```

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
opencc-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
`Vocal10n-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
`Vocal10n-prebuild/GPT-SoVITS/` (entire directory)	`vendor/GPT-SoVITS/`	~3GB+ with pretrained models
`Vocal10n-prebuild/live-translation-pipeline/reference_audio/`	`reference_audio/`	Sample audio files
`Vocal10n-prebuild/personal-logger/context_gaming.txt`	`knowledge_base/`	Gaming terminology for RAG

Key Architecture Decisions

1. **Event-driven pipeline** — Keep the pub/sub `EventDispatcher` pattern from prebuild. It cleanly decouples STT → LLM → TTS.
2. **PySide6 signals** — Qt signals/slots replace Gradio's polling. Real-time text updates via `QTextEdit.append()` with custom signals from worker threads.
3. **Worker threads** — STT, LLM, and TTS each run in `QThread` workers. The coordinator manages lifecycle and event routing.
4. **Config-driven** — Single `default.yaml` config file controls all parameters. UI changes write back to config. Config is the source of truth.
5. **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()`.

目前来看我们已经到了第三步，也就是测试 STT这一块。它告诉我不要用 RAG,直接把信息写在 initial prompt 里面（不过我确实还是怀疑这玩意的性能问题，所以一会再进行测试。比如第一行是能检测出来，最后一行的词是不是就该忘了。）在此之前，不得不说这新款 UI 确实好看，并且在实时更新之类的这一块完全没有任何问题。测试完 STT 之后，我们让它继续进行下一步的整合，也就是融合 LLM进来。我们之前搭建了一套基于 Qwen3的 LLM 翻译器，接下来我会让它从以前的文件里面找到这个 build过的版本，这样就不用从零 build一个让 Qwen3 跑起来的框架了。

然而这玩意跑了一下结果翻译一条竟然需要五秒时间，一眼就是没用 Cuda加速。

我们目前似乎遇到了一些问题，遇到了之前一样的问题，也就是 Qwen3不知道怎么的没法用 CUDA 支持，又要自己从头 build 了。我以为它能够自己找到老版本做好的那个文件，但似乎它并没有理解我的意思，我们一会看看怎么个事。现在是 19:10

19:27, 它似乎找到了一个 ggml-cuda.dll和 llama.dll, 然后我问了关于 linux端，它说 linux端用的是 .so 文件。所以我猜大概率也要重新 build一个? 另外就是我用的 GGUF是 mradermacher/Qwen3-4B-Instruct-2507-GGUF版本的，而不是 lmstudio版本的。我需要搞清楚这俩有啥区别。问了 Gemini， LMStudio 版本的大概是为了 LMStudio特制的，而 mradermacher的则更为通用。本质上来讲区别不是特别大（大概）。

我们把 Context这块的问题搞明白了，目前我们似乎是有 rag系统的，只不过加载了 200个 term 在初始提示词里。所以或许整体识别速度会更快一点。另外，我让它开始更改 UI 上的一些问题，主要是文字超出了按钮的显示范围导致没法完全显示成功。现在是 19:52

破案了，我们让他用 unicode编码的符号字符就完事了。另外顺便让它改一下 UI 的总体结构，毕竟现在很多按钮都重合到一起去了。

时间来到了 22:20, 我们成功再一次跑通了从我说话到系统说话的全流程，接下来就是完善 UI 和提升效率这块的问题了。首先我们现在的问题是 TTS输出非常之慢，达到了 9秒左右，而之前的老系统大概是 2秒。在花了一点时间解决之后，现在我们的 TTS系统可以达到惊人的 25毫秒延迟，史无前例的低。

时间来到了 22:56，我现在才知道这 25ms是 http响应时间，并非 TTS□□□□□□□ TTS响应时间似乎是500ms左右，但现在被搞成了交传并非同传，我们正在解决这个问题。

现在开始这玩意被越改越慢了。 Claude
意识到老架构比新架构强，但现在还是在老架构上打补丁。导致目前我们 TTS的响应水平奔着 18
秒去了，而且越来越慢。终于在一番折腾下，我们在 23:46
基本解决了之前累积着不出声音的问题。不过现在的问题是我第一句话如果太长，他也要等我第一句话说完才
开始说话。

时间一转眼来到了 1:05，现在是

02.10

目前我们已经解决了实时语音输出的问题，总体上来看暂时还行，后面我们再解决识别准确率和延迟的问题。
现在的首要任务是把整个流程跑通。 OBS
现在字幕的效果不错，我给它加上了延迟消失，让看视频的用户阅读起来更方便。按照现在的进度来看，我们
基本可以告一段落了，后面主要就是识别准确性，降低延迟以及模型训练的事情了。照着目前的测试结果来看
，我念了一段新闻， STT大概平均 1.3-1.5秒， LLM翻译 0.5秒，但 TTS要 8秒开外。

目前接近 2:00，我们打算让它写出一个 RAG的标签页以及一个训练的标签页之后就收工。明天可以再从前面的
STT消除幻觉开始，然后提升 TTS的转换速度。

2.11

休息了一天，现在是 0:28。偶然发现一个 AI女友项目 (文档详见这里_____)，这玩意就是 TTS+LLM+STT还多一个
live2d展示窗口，我寻思这不就是我想要做的东西吗？我真的服了。正好这也是个 MIT
协议开源项目，我倒是可以让 Copilot参考这个项目回头多集成一些不同的 TTS/STT引擎进去。

??

03.09~03.15

3.11

没错，我休息了一个月，把重心转向了一些个人项目。今天是 GDC的第三天，星期三。 Max
下午开始问我什么时候跟他约个开会时间，我一想那不就讲讲现有的这些东西么，结果发现竟然过去一个月了
还没什么进展。更进一下上一次我们说倒 Qwen3.5的模型，那玩意只是一个 TTS模型，并不能做 STT
方面的东西。另外今天跑了一下测试，发现这个英文语音 >
英文文本现在不能输出中文，这很明显不对劲。因为同一个项目用的也是 GPTSovits
但就是可以用正常的中文进行输入 /输出。所以首先我们要解决这个问题，然后再解决一下模型的准确性问题。

改到 20:40发现是之前加 Qwen3TTS的时候把这俩做到一起去了，相当于 GPT是一台机器， Qwen3
是一台机器，同时控制 GPT和 Qwen的两个开关分别存在。只不过 Qwen和 GPT
的开关都连到了某个中央开关，因此开任何一个开关都可以打开这个中央开关，然后这个中央开关会同时打开
两个机器。现在要做的就是让 Qwen的开关只控制 Qwen, GPT的开关只控制 GPT。然后 Qwen/GPT
只能二选一启动就完事了。

[illegible] 21:30

[illegible]

[illegible] “Patreon[illegible] [illegible] Patreon

[illegible] [illegible] [illegible] [illegible]

[illegible] “[illegible]” [illegible]”

[illegible] 23:30[illegible] TTS[illegible] Qwen3.5 TTS

[illegible] GPT-SoVITS[illegible]