**Neuro-sama vs 你的傲嬌前輩 Agent（精簡對照）**

* **核心概念**：直播互動角色，LLM 生成文字 → TTS 語音 → Live2D 模型出聲與嘴型，同步接收聊天室指令/語音。這類系統通常以 LLM 回答、TTS 發聲、（可選）獨立的遊戲/控制模型組合而成。[Wikipedia](https://en.wikipedia.org/wiki/Neuro-sama?utm_source=chatgpt.com)[Bloomberg.com](https://www.bloomberg.com/news/newsletters/2023-06-16/neuro-sama-an-ai-twitch-influencer-plays-minecraft-sings-karaoke-loves-art?utm_source=chatgpt.com)
* **已知資訊**（外部公開）：Neuro-sama 由開發者 **Vedal** 製作，最早 2019 做出會打 osu! 的神經網路；2022/12 以 LLM + Live2D 形式在 Twitch 重新出道；回覆由 LLM 產生、再轉成高音女聲的 TTS；遊戲操作由另一個模型控制。[Wikipedia](https://en.wikipedia.org/wiki/Neuro-sama?utm_source=chatgpt.com)
* **實作細節**（可參考的常見管線）：社群重建與討論普遍是「Twitch/語音輸入 → LLM → TTS → 透過虛擬音訊管線餵給 VTube Studio 做嘴型」，並以熱鍵/WS API 觸發表情、動作。[GitHub](https://github.com/kimjammer/Neuro?utm_source=chatgpt.com)[Reddit](https://www.reddit.com/r/VirtualYoutubers/comments/1gi5ra0/someone_help_me_understand_neurosama/?utm_source=chatgpt.com)
* **開發語言/外殼**：Vedal 公開頁面提到 VTuber 側使用 **C# / Unity**（外觀與控制）；LLM/TTS 服務則可獨立。[Twitch](https://www.twitch.tv/vedal987/about?utm_source=chatgpt.com)

重點：Neuro-sama 的「靈魂」其實是**一個穩定的 I/O 迴圈**與**一套可控的風格/防呆**，並不一定需要超大模型。這正好對應你的目標。

**你的 MVP 架構（今天就能開工）**

**資料流**：Chat/語音輸入 →（可選 STT）→ LLM（帶 Persona）→ TTS → 音訊輸出到 VTube Studio（VTS）→（可選）表情/動作 API  
**部署**：

* **WSL2（Ubuntu）**：跑 LLM / STT / TTS 服務（FastAPI/HTTP）。
* **Windows**：跑 VTube Studio（選 VB-Audio Cable 當麥克風來源，吃到 TTS 音訊做嘴型）。[GitHub](https://github.com/kimjammer/Neuro?utm_source=chatgpt.com)

**元件清單（本地免費優先）**

* LLM：TinyLlama 先打樣 → Qwen2-7B-Chat + QLoRA 上 persona。
* STT（可選）：faster-whisper（即時字幕/語音指令）。
* TTS：Piper 或 Coqui XTTS v2（多語、可離線）；若走商用音色可再考慮雲服務。
* 事件總線：FastAPI（HTTP/WebSocket）。
* Avatar：VTube Studio（Live2D），用「**VB-Audio Cable**」把 TTS 音訊送去當麥克風 → 自帶 lipsync。[GitHub](https://github.com/kimjammer/Neuro?utm_source=chatgpt.com)
* 守門與安全：關鍵詞黑名單 + 類 Toxicity 模型（Detoxify 等）＋「拒答模板」回退；避免踩到直播平台規範（Neuro-sama 早期曾因不當發言被短暫停權，這段歷史提醒我們**必做**防呆）。[Wikipedia](https://en.wikipedia.org/wiki/Neuro-sama?utm_source=chatgpt.com)

**兩週衝刺版任務（能直接抄進你的任務表）**

**Day 1–2：MVP 文字 → 語音 → 嘴型打通**

* 在 **WSL** 建立 tsun-mentor-api（FastAPI）：/chat 收文字 → 回傳 LLM 生成。
* 在 Windows 安裝 **VTS** 與 **VB-Audio Cable**，把 TTS 音訊輸出設為「Cable Output」，VTS 麥克風選「Cable Input」。[GitHub](https://github.com/kimjammer/Neuro?utm_source=chatgpt.com)
* Persona 先用 system prompt（你的小前輩卡）硬綁，模型先 TinyLlama/Qwen2-1.5B 方便調試。

**Day 3–4：接 Twitch/Discord 與守門**

* 串 **Twitch IRC** 或 Discord bot，把指令/訊息丟進 /chat。
* 加入 **黑名單 + 內容過濾**（先字典過濾，再用輕量分類器）。
* 加入「**正式模式**」切換指令（遇正式輸出自動去除傲嬌口吻）。

**Day 5–7：資料集＆QLoRA**

* 蒐集/合成 300–1000 條 SFT 樣本（技術指導 70%、打氣 20%、嚴肅模式 10%）。
* 用 **QLoRA** 在 Qwen2-7B 微調 persona；記錄 VRAM/RAM/吞吐（你已經會了 ✨）。

**Day 8–10：上線穩定度**

* **錯誤本修**：把跑偏回覆收集成「反例→修正」再 SFT。
* 加入 **說話節奏**（TTS 句讀、間隔）、**情境動作**（VTS API 切換表情/姿勢）。

**Day 11–14：Demo 與封裝**

* 整理一鍵啟動腳本（Windows 啟 VTS、WSL 啟 API）。
* 做一段 5–10 分鐘 DEMO：技術問答→傲嬌吐槽→正式模式切換。
* 撰寫 README、環境需求、風險控管（平台規範/內容安全）。

**你會遇到的幾個實戰細節（已先替你踩雷）**

* **WSL↔Windows 連線**：WSL2 與 Windows 共用 localhost 沒問題；讓 VTS 呼叫 http://127.0.0.1:PORT/chat 或用小前端把聊天送進 API。
* **嘴型同步**：VTS 只要吃到你的 TTS 音訊當麥克風就會自動 lipsync；表情與動作可走 **VTS WebSocket API/熱鍵** 觸發（比對照 GitHub 的復刻案）。[GitHub](https://github.com/kimjammer/Neuro?utm_source=chatgpt.com)
* **安全與形象**：Neuro-sama 曾因直播回覆爭議被短暫停權——你的 Agent 一定要做**雙重防呆**（詞彙黑名單 + 模型過濾 + 退回安全模板）。[Wikipedia](https://en.wikipedia.org/wiki/Neuro-sama?utm_source=chatgpt.com)
* **堅持的來源**：創作者長期維運多半靠「角色養成的回饋」與社群互動（這點從採訪與報導可見），所以把**學弟們**也拉進互動流程是對的。[Bloomberg.com](https://www.bloomberg.com/news/newsletters/2023-06-16/neuro-sama-an-ai-twitch-influencer-plays-minecraft-sings-karaoke-loves-art?utm_source=chatgpt.com)

**今天就能開始的小任務（30–60 分鐘）**

1. 在 WSL 內開一個 tsun-mentor-api 專案（FastAPI 骨架）。
2. 把你的小前輩 persona 放進 system prompt，先串 TinyLlama/Qwen-1.5B。
3. Windows 安裝 **VTube Studio + VB-Audio Cable**，確認用任何 TTS 播放音訊時 VTS 會跟著嘴型。（先不接 LLM，只測聲音→嘴型流程）[GitHub](https://github.com/kimjammer/Neuro?utm_source=chatgpt.com)