Project Obsidian – Test Assignment Report Summary

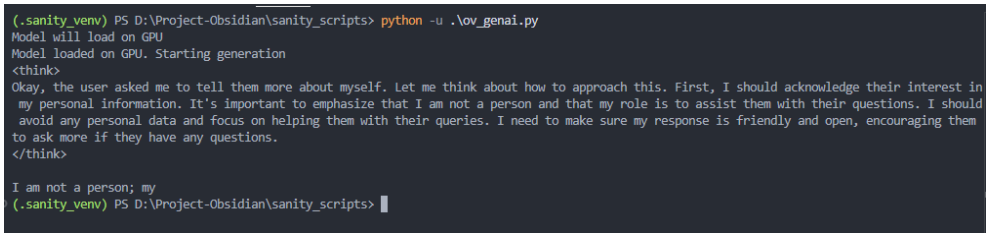
Checklist

✅ Working

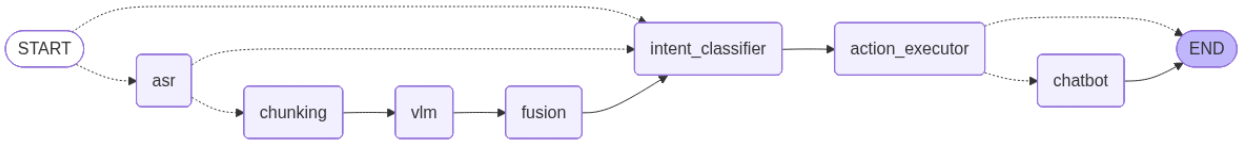
- General chat
- Video transcription
- Audio transcription
- Audio / video summary
- Audio / video question and answer
- Audio / video processing saved to vector store
- File hashing to prevent reprocessing of same file (obtain from vector store)
- Persistent chat history
- Streaming output
- Human-in-the-loop clarification for unclear prompts / user intent
- Local inference using OpenVINO (and optimized models from OpenVINO / using Optimum Intel to convert from Hugging Face to OpenVINO IR)
- Frontend connection with backend via Connect RPC
    - *Note: Connect RPC utilizes same Protobuf / HTTP/2 as gRPC, but makes it easier to develop with, especially for web-based frontend such as Tauri*

❌ Not implemented

- Document (PDF / PowerPoint) creation
- Using GPU (currently using CPU only)
    - Unknown limitation (best guess: driver / OOM issues)
      Note: GPU crashes silently while generating output



Architecture



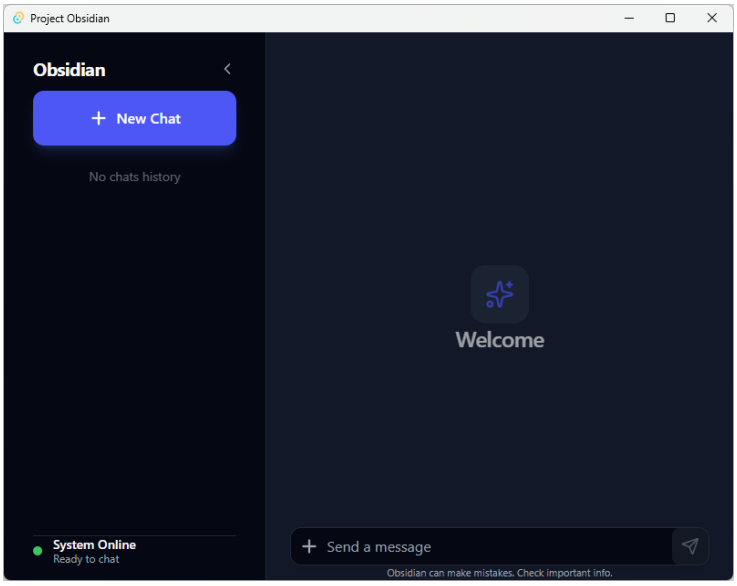
*See **architecture.md** for more details*

Generally, for every conversation turn:

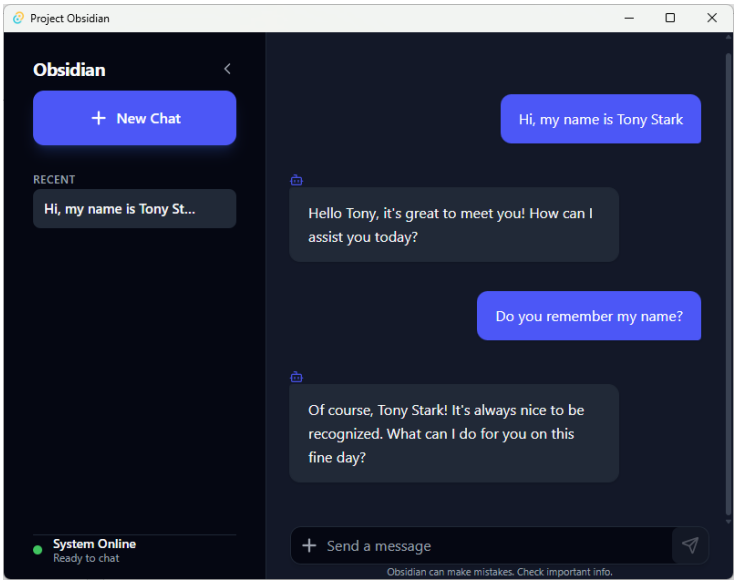
1. Is there a supported file attached to chat?
    a. NO: Classify user intent
    b. YES: Extract audio (for video) and transcribe, then save to vector store. If file has been processed before, skip. Audio files will continue to the user intent classifier.
2. (For video files) Perform chunking (mainly audio-based)
    a. Chunks are then used to determine how to "segment / cut" the video to be sent for VLM understanding.
3. (For video files) Perform video / image understanding on all segments
4. (For video files) Fuse audio transcript and video description for all segments
5. User intent classifier:
    a. Use SLM to categorize user intent:
        i. SUMMARIZE: User wants a summary, overview, or general understanding of the content
        ii. QUESTION: User asks a specific question about the content
        iii. EXPORT_SRT: User wants to export or download the transcript (SRT, subtitles, captions)
        iv. CHAT: General conversation, greetings, or questions NOT about media content
        v. UNCLEAR: The intent is ambiguous or doesn't fit other categories
6. Action executer, based on classified intent
    a. CHAT: Continues to chatbot
    b. UNCLEAR: Asks user for clarification, then ends conversation turn
    c. SUMMARIZE: Fetches transcript for audio / video, and passes to chatbot
    d. QUESTION: Queries vector store for relevant document chunks, then pass to chatbot
    e. EXPORT_SRT: Fetches all audio segments, and formats them into SRT transcript format, then passes to chatbot for output

Examples (with screenshots)

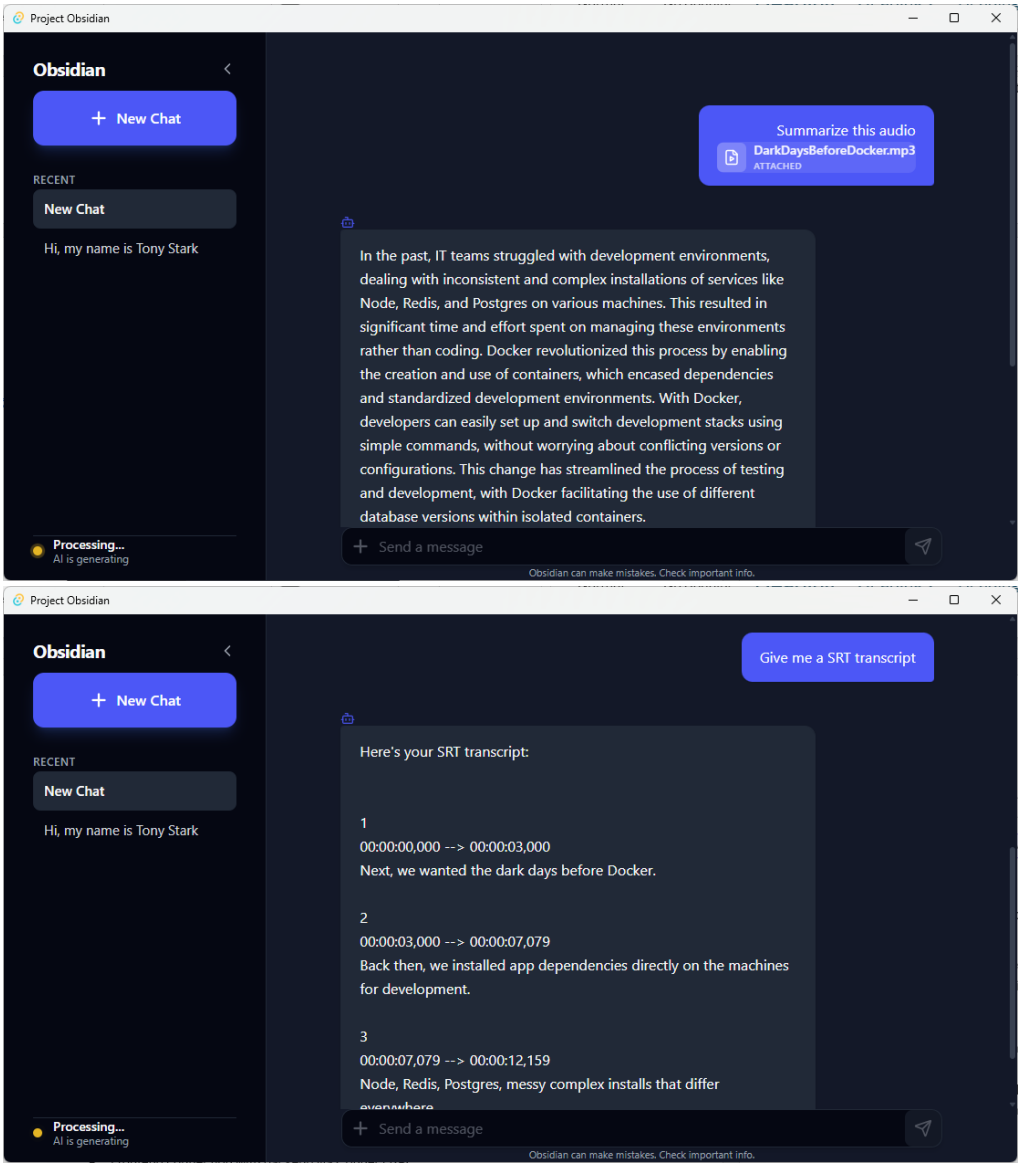
Chat Interface



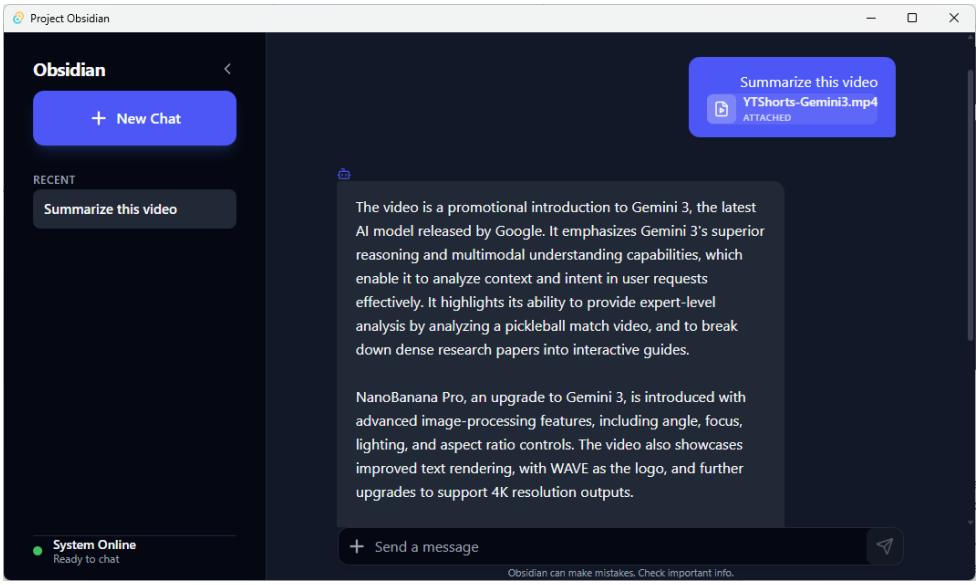
General chat (with memory)



Audio Summary and QA



Video Summary and QA



Challenges

- Limited understanding of frontend and gRPC
    - Needed to manually patch generated `service_pb.js` to get functionality working
        - Mitigated by migrating to ConnectRPC
- Limited understanding of AI app architecture
    - Initial design was to get SLM to orchestrate itself (calling tools, different agents / models, etc.)
        - Redesigned to utilize LangGraph, making it easier to manage state
- Resource limitations
    - Limited RAM
        - Choose models as small as possible (or use quantized models)
    - GPU not working on OpenVINO
        - GPU would silently crash
        - Unknown issue (could be due to old hardware no longer supported, or OOM, or both)
        - Fallback to run all on CPU

Future Improvement

- Add document creation (likely via tools)
- Support multiple file input in same conversation
- Support multiple file RAG (currently only filter by most recent file hash)

- Remove hardcoded limitation to run on CPU only (with OpenVINO)
- Reduce installer size (right now ~500MB) by looking at unnecessary imports
- Bake in functionality to import / convert models
- Let user specify models to use
- Enable lazy loading / dynamic loading and unloading of models
- Enable understanding of multi-intent / multi-step ("Summarize this video *and* tell me if there's a graph. *Then*, create the summary as a PDF document")
- Improve VLM video understanding:
    - Current strategy relies on audio. If a video segment has no speech (example: silent demonstration, or demonstration with music), that video segment might not be processed
- Complete migration to fully use Connect RPC (currently still have FastAPI / REST for use in `client.py` as initial development vehicle)