Local Agentic AI — Phase■1 (MVP) Documentation

Windows 11 + WSL2 • Offline■first • Least■Privilege • Audited Tooling

1. Executive Summary

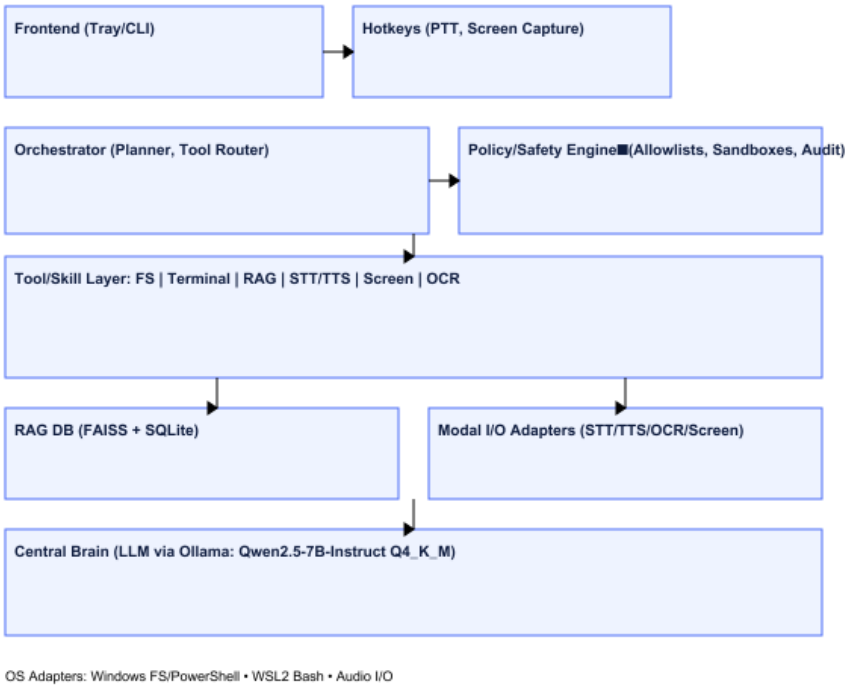
This document specifies, explains, and operationalizes the Phase■1 MVP of a local, multimodal, agentic AI system. The MVP enables push■to■talk voice commands that summarize a sandboxed local file with inline citations and speaks the result. It also provides screen capture with OCR, a RAG■lite index, allowlisted terminal commands, and comprehensive JSON audit logs. The target environment is Windows 11 (PowerShell 7) with WSL2 (Ubuntu 24.04).

Success Criteria (Definition of Done)

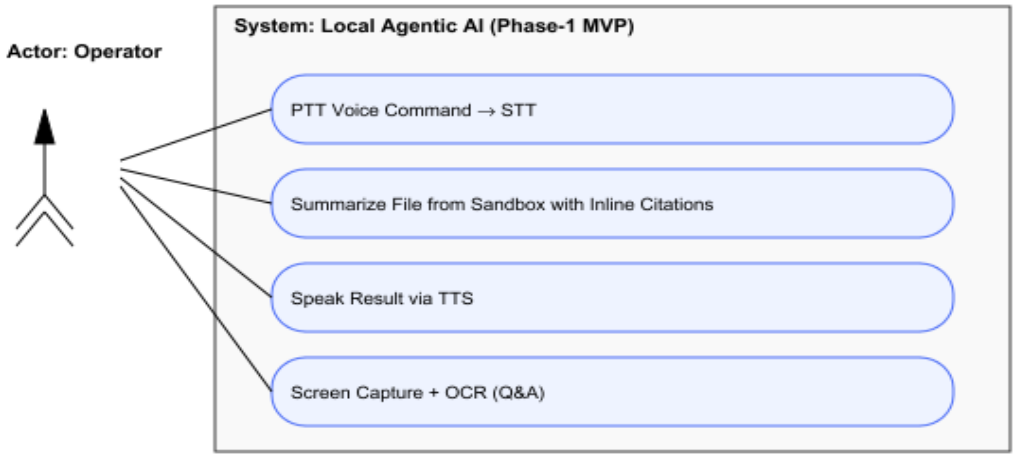
• Voice → STT → Planner → FS/RAG → Response → TTS. • Summary includes inline citations (path:line). • Screen capture + OCR accessible via hotkey. • All tool calls are logged with tamper■evident audit entries. • Runs fully offline after initial model downloads.

2. Reference Architecture

High■level data/control paths across frontend, planner, tools, RAG, and the LLM runtime.

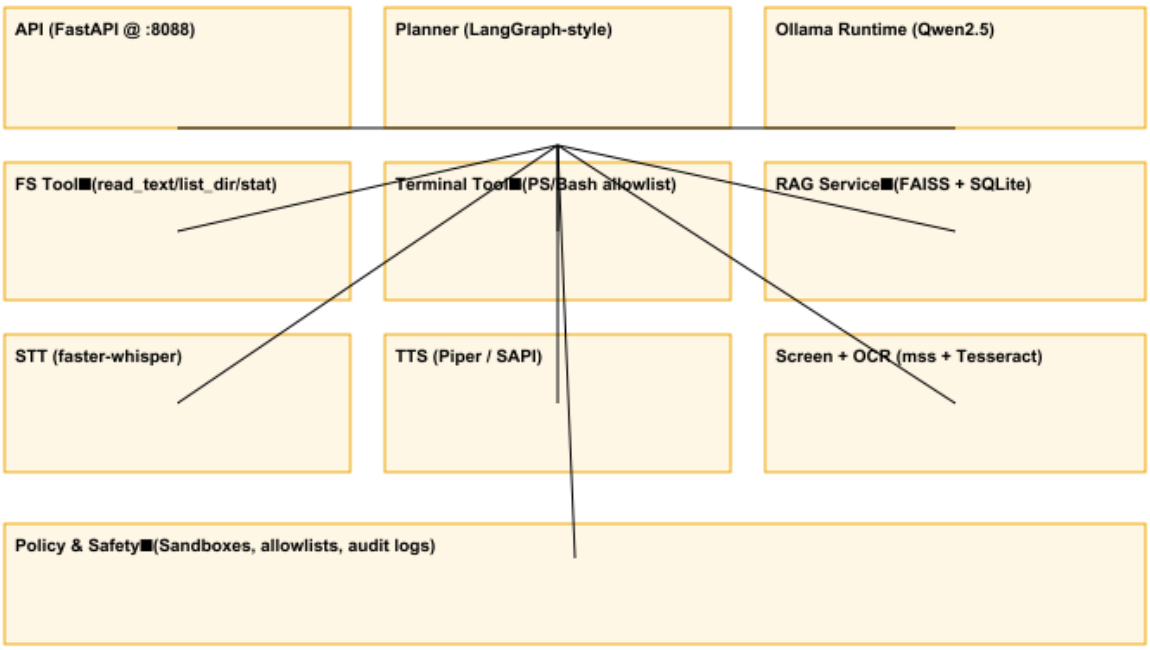


3. Primary Use Cases & Actors



Actors: Operator (local user).
Use Case 1: PTT Voice → Summarize a specified file from sandbox; include (path:line) citations; speak result.
Use Case 2: Screen Capture + OCR → Answer questions about visible text/UI.
Use Case 3: Terminal (allowlist) → Read■only diagnostic commands (e.g., *git status*), no destructive ops in Phase■1.

4. Component View



The API (FastAPI) exposes tool endpoints consumed by a planner that coordinates the LLM (Ollama/Qwen2.5). Tool adapters provide filesystem, terminal, RAG, STT/TTS, and screen/OCR functionality. A policy layer enforces sandboxes, allowlists, and auditing.

5. Codebase Structure (Phase■1)

```
C:\Agent\
  bin\            # operator scripts (agent.ps1, hotkeys, reindex)
  config\         # schemas, defaults, allowlists
  models\         # model caches (RAG, Piper voice, etc.)
  logs\           # JSON logs + artifacts (wav, screenshots)
  audit\          # tamper■evident audit JSONL
  ingest\         # drop■in folder for documents to index
  src\
    api\main.py               # FastAPI app
    brain\runtime.py          # Ollama client wrapper
    brain\planner.py          # MVP planner (summarize file flow)
    common\logging.py         # structured logging + audit writer
    policy\safety.py          # sandboxes, traversal checks, allowlists
    tools\fs.py               # read_text, list_dir, stat
    tools\shell.py            # PS/Bash allowlist execution
    tools\rag.py              # FAISS+SQLite + chunker
    tools\stt.py              # faster■whisper wrapper (int8, CPU)
    tools\tts.py              # Piper (CLI) + SAPI fallback
    tools\ocr.py              # Tesseract OCR
    tools\screen.py           # mss capture
  tests\           # unit + e2e harness
  docs\            # install/run docs
```

Key Modules

Planner: Deterministic routing for MVP: detect 'summarize file' intent, enforce policy, use FS and RAG, compose answer with citations, speak via TTS.
Policy: Path normalization, sandbox enforcement, traversal denials, terminal allowlists, audit IDs.
RAG: FAISS (inner■product) for dense retrieval using BGE small; SQLite for metadata; chunker based on line counts.

6. Security & Safety Controls

• Least■privilege: filesystem restricted to configured sandboxes; terminal limited to allowlisted commands. • No write ops in Phase■1; destructive categories disabled entirely. • Hotkey■gated capture/STT; no continuous listening; no background scraping. • Audit trail: per■tool JSONL entries with timestamp and payload hashes.

7. Detailed Flows

Flow A — Voice → **Summarize File** → **Speak:** PTT audio capture → STT (faster■whisper) → Planner intent → FS read + optional RAG highlight → LLM summary with inline (path:line) → TTS.

Flow B — Screen OCR Q&A;: Screen capture (mss) → OCR (Tesseract) → Context passed to LLM → Answer; citations when from file content.

Flow C — Terminal (Allowlist): Planner requests command; policy validates against allowlist; API executes and returns truncated stdout; no writes allowed.

8. Minimal APIs (FastAPI Endpoints)

```
POST /chat                            -> planner intent; routes tools; returns {ok, text}
POST /tools/fs {op, path}             -> read_text | list_dir | stat
POST /tools/shell {shell, cmd, cwd?}  -> PS/Bash read-only allowlist
POST /tools/rag/query {q, top_k?}     -> [{path, line, snippet, score}]
POST /tools/screen/capture            -> {image_path, size}
POST /tools/ocr {image_path, lang, psm}-> {text}
POST /tools/stt {audio_path?, size, language} -> {text}
POST /tools/tts {text}                -> {out_path}
```

9. Configuration (Excerpt)

• runtime: engine=ollama, model=qwen2.5:7b-instruct, quant=q4_K_M, context=8192, offload_layers≈20. • safety: terminal_mode=allowlist, require_confirm_destructive=true, hotkeys for PTT and screen. •

sandboxes: absolute paths for read operations. • rag: BGE small embeddings (CPU), FAISS index dir, chunk size/overlap. • logging: INFO; JSONL logs in C:\Agent\logs.

10. Deployment & Operation (Quickstart)

1) Create folder tree under C:\Agent. 2) Python 3.11 venv; install deps (FastAPI, faster-whisper, faiss-cpu, sentence-transformers, pypdf, mss, pillow, pytesseract, sounddevice, soundfile, keyboard). 3) Install Tesseract via Chocolatey; ensure 'eng' and 'fas' language packs. 4) Install Ollama; pull qwen2.5:7b-instruct (Q4_K_M auto). 5) Place a Piper voice ONNX; set PIPER_VOICE env var (SAPI fallback exists). 6) Edit config.json for sandboxes. 7) Reindex documents via bin/reindex.py. 8) Start agent with bin/agent.ps1 start. 9) Press Ctrl+Space and request a summary of a specific sandboxed file (full path). 10) Verify logs and audits.

11. Observability & Auditing

Structured JSON logs in logs/agent.jsonl; Audits per day in audit/YYYY■MM■DD.jsonl with audit_id, tool, hashed payload, duration, and outcome.

12. Performance Notes

• GTX■960 (~4 GB) handles 7B Q4_K_M with partial offload; expect ~2–3 tok/s; CPU fallback ~1 tok/s. • STT tiny/base int8 near real■time on CPU. • OCR per 1080p region typically <1s; whole screen a few seconds. • Keep context ≤8k for stability on 12 GB RAM.

13. Testing Strategy (Phase■1)

Unit: safety checks (traversal, allowlists), RAG round■trip, OCR smoke. E2E: PTT voice → summarize todo.md with citations → TTS WAV emitted. Chaos: Missing GPU/audio device; index corruption; denied sandbox.

14. Risks & Mitigations

• VRAM pressure → drop offload layers or use CPU fallback. • Noisy mic → fix language, use base/tiny; enforce PTT. • OCR on dark UI → capture region; adjust PSM. • Path traversal → strict normalization + sandbox checks. • Command injection → allowlist head-only command parsing.

Appendix A — Representative Code Snippets

```
# src/tools/fs.py (read_text excerpt)
if deny_traversal(path) or not is_in_sandboxes(path, sandboxes):
    res={"ok":False,"error":"path_not_allowed","path":path}
    audit("fs.read_text", {"path":path}, res); return res
with open(normalize_path(path), "r", encoding="utf-8", errors="ignore") as f:
    txt = f.read()
res = {"ok":True,"bytes":len(txt.encode()),"text":txt}
audit("fs.read_text", {"path":path}, {"ok":True,"bytes":res["bytes"]})
```

```
# src/policy/safety.py (allowlist check excerpt)
def check_ps_allowlist(cmd: str, allow: List[str]) -> bool:
    if _BAD.search(cmd): return False
    head = cmd.strip().split()[0].lower()
    return any(head.lower() == a.lower() for a in allow)
```

```
# src/brain/planner.py (intent route excerpt)
m = SUMMARIZE_PAT.search(text or "")
if m:
    path = extract_path_from(text)  # quoted or after 'in'
    if not path: return {"ok":False,"error":"no_path_found","hint":"Provide full path or quoted filename."}
    return summarize_file(flow, path, sandboxes, rag, llm, voice_model)
```

Contact & Operator Notes

This Phase■1 MVP is designed for rapid, reliable local operation with strong guardrails. Extend to Phase■2 by enabling controlled write ops with confirmations, hybrid retrieval (BM25+dense), reranking, richer planning, and wake■word support.