**FINAL DEV STACK (confirmed)**

**Frontend / Overlay**

|  |  |  |
| --- | --- | --- |
| **Layer** | **Tool** | **Purpose** |
| **Language** | C# (.NET 8) | Core logic and orchestration |
| **UI Framework** | **WPF** (Windows) | Native, responsive overlay UI |
| **Web UI Layer** | HTML / CSS / JS (via WebView2 or Blazor) | Clean styling + animation |
| **Audio Capture** | **NAudio** | Capture mic/system audio |
| **Realtime Updates** | **SignalR** | Stream AI responses to UI |

Reasoning: WPF gives native-level performance. WebView2 (HTML/CSS/JS) makes it easier to design sleek interfaces.

**Middleware / Backend (Local services)**

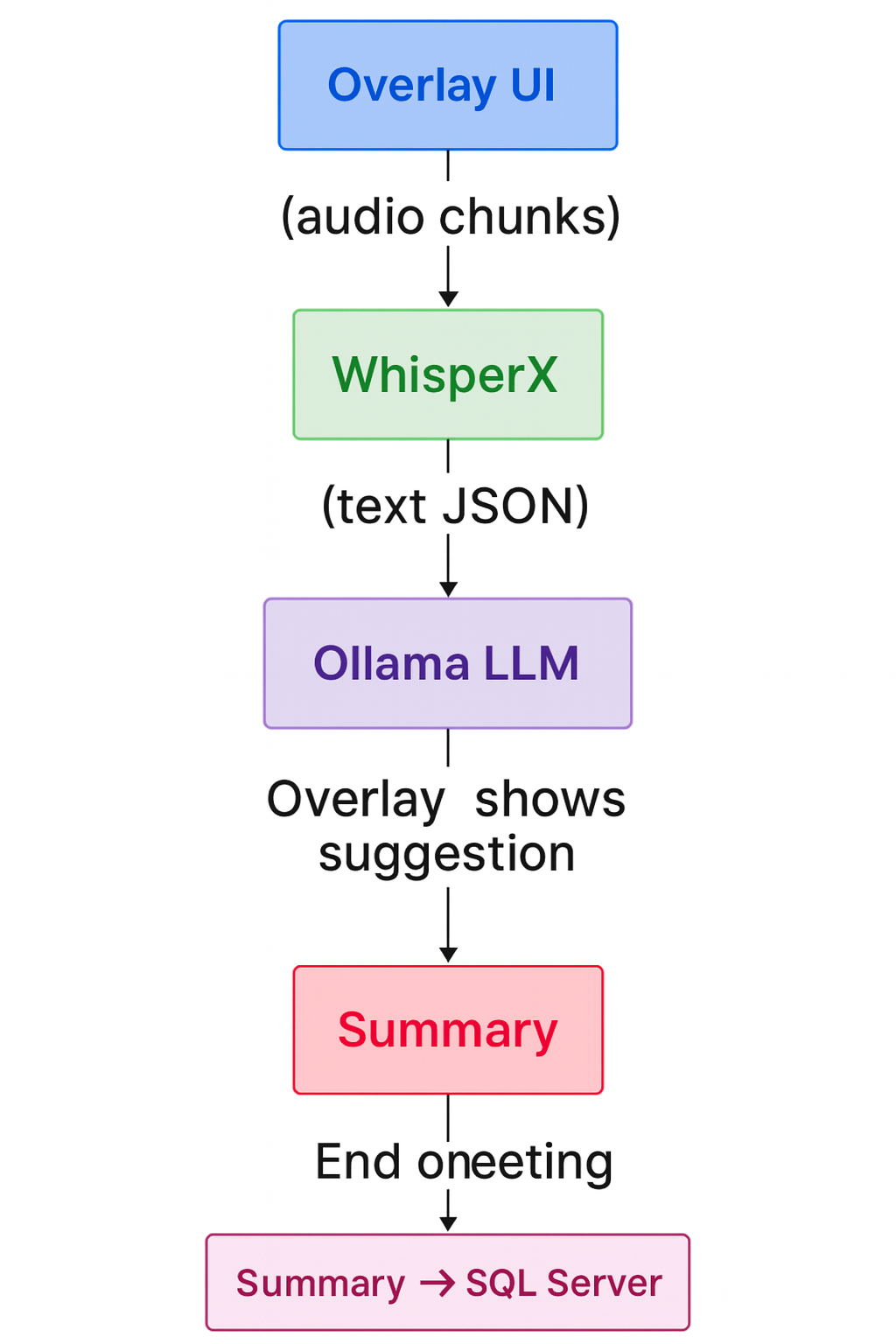
|  |  |  |
| --- | --- | --- |
| **Layer** | **Tool** | **Purpose** |
| **Whisper Service** | Python + WhisperX | Speech-to-text + speaker diarization |
| **AI Orchestrator** | C# or Python service | Coordinates transcript → context → LLM |
| **Realtime Cache** | Redis | Stores rolling transcripts, meeting context |
| **LLM Engine** | Ollama (running Phi-3 Mini, Gemma 2, or Llama 3 8B) | Real-time response generation |
| **Summary Engine** | Claude 3.5 or GPT-4o-mini (cloud fallback) | End-of-meeting summary generation |
| **Database** | SQL Server | Stores final summaries, transcripts, metadata |

Reasoning: This split lets you use Whisper and Ollama locally for speed, Redis for live memory, and SQL only at the end.

**Infrastructure / Deployment**

|  |  |
| --- | --- |
| **Component** | **Tool** |
| Local containerization (optional) | Docker (for Redis, Whisper, Ollama) |
| Logging / Monitoring | Serilog or Seq |
| Package management | NuGet (C#), pip (Python) |
| Source control | GitHub / Azure DevOps |
| CI/CD (deployment) | GitHub Actions or Azure Pipelines |

**DATA FLOW RECAP**

****

**DEVELOPMENT PLAN**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Deliverable** | **Time Frame** |
| **PHASE 1 – Foundation Setup** | **Goal:** Project environment & skeleton ready. **Deliverables:**   * Set up .NET solution + WPF overlay UI project. * Add WebView2 container (for HTML/CSS/JS front-end). * Create placeholder panels for:   + Mic control   + Real-time suggestions   + Meeting context input * Create basic local REST endpoint (ASP.NET or Python FastAPI). * Setup GitHub repo + CI pipeline.   **Output:** A blank overlay app that opens, captures mic, and displays static data. *“Hello, World” but for your meeting assistant.* | **(Week 1–2)** |
| **PHASE 2 – Audio Capture + Whisper Integration** | **Goal:** Real-time transcription pipeline working. **Deliverables:**   * Integrate **NAudio** to capture audio stream. * Send audio chunks to **WhisperX** microservice (Python). * Receive transcribed text with speaker tags. * Store transcript in Redis (e.g., meeting:transcript key). * Display live transcript feed in overlay.   **Output:** You speak → overlay shows live text in real time. | **(Week 3–4)** |
| **PHASE 3 – LLM Integration + Real-time Suggestions** | **Goal:** AI starts giving in-meeting responses. **Deliverables:**   * Set up **Ollama** locally and connect via REST API. * Feed LLM recent transcript + meeting context from Redis. * Generate short suggestion text (1–2 sentences). * Display suggestions in overlay instantly via SignalR/WebSocket. * Add simple toggle for “Cloud mode” (Claude/GPT for fallback).   **Output:** App listens → understands context → shows live AI suggestions. | **(Week 5–6)** |
| **PHASE 4 – Summary & Storage** | **Goal:** End-of-meeting summary + SQL storage. **Deliverables:**   * + Collect full transcript from Redis at meeting end.   + Send to cloud LLM for final summary generation.   + Store:     - Summary text     - Transcript (optional)     - Meeting metadata (time, duration, tags) → into SQL Server.   + Add “View summary” UI panel in overlay.   **Output:** After meeting, user gets summary saved in database and readable in UI. | **(Week 7)** |
| **PHASE 5 – Polish, Optimize, and Package** | **Goal:** Ready-to-deploy MVP build. **Deliverables:**   * Add UI polish (icons, loading indicators, fade animations). * Optimize Redis memory eviction (old transcripts auto-expire). * Package app with installer (MSIX or ClickOnce). * Final test: simulate meeting with real audio, confirm flow. * Create a one-page documentation / README.   **Output:** Installable, deployable **MVP meeting assistant**. | **(Week 8)** |

**Summary Timeline (Compact View)**

|  |  |  |
| --- | --- | --- |
| **Week** | **Milestone** | **Deliverables** |
| 1–2 | Setup & UI | Overlay base app, environment ready |
| 3–4 | Audio + Whisper | Live transcription pipeline |
| 5–6 | LLM + Redis | Real-time suggestions + caching |
| 7 | SQL & Summary | Final storage, meeting summaries |
| 8 | Polish & Package | Optimization, packaging, testing |

**Post-MVP Ideas (Future)**

Once MVP works:

* Add **meeting summaries dashboard** (past meetings).
* Add **“Interview Mode”** — LLM tailors answers to interview context.
* Add **speech playback** for Whisper output.
* Add **multi-platform** via .NET MAUI.
* Add **analytics** — measure AI response speed or user satisfaction.

**Project Folder Layout (Production-Ready)**

MeetingAI/

├── src/

│ ├── OverlayApp/ # C# WPF frontend

│ ├── BackendAPI/ # ASP.NET Core API

│ ├── WhisperService/ # Python service

│ ├── RedisConfig/ # Hot storage setup

│ ├── SqlServerDB/ # Database schema + EF models

│ └── OllamaConnector/ # Handles local LLM calls

├── tests/

│ └── UnitTests/

└── docs/

└── Architecture.md

**✅ TL;DR – The Deliverable Plan**

| **Category** | **Stack** | **Notes** |
| --- | --- | --- |
| Frontend | C# + WPF + HTML/CSS/JS | Overlay app |
| Speech Processing | Python + WhisperX | Real-time transcription |
| Realtime Memory | Redis | Hot data layer |
| AI Reasoning | Ollama + Phi-3 | Local suggestions |
| Final Storage | SQL Server | Summaries + transcripts |
| Cloud Option | Claude / GPT | High-quality summaries |
| Target MVP | 8 weeks | Deployable Windows app |