Project Documentation: The Agentic AI Interviewer

Submitted by Team Nazi

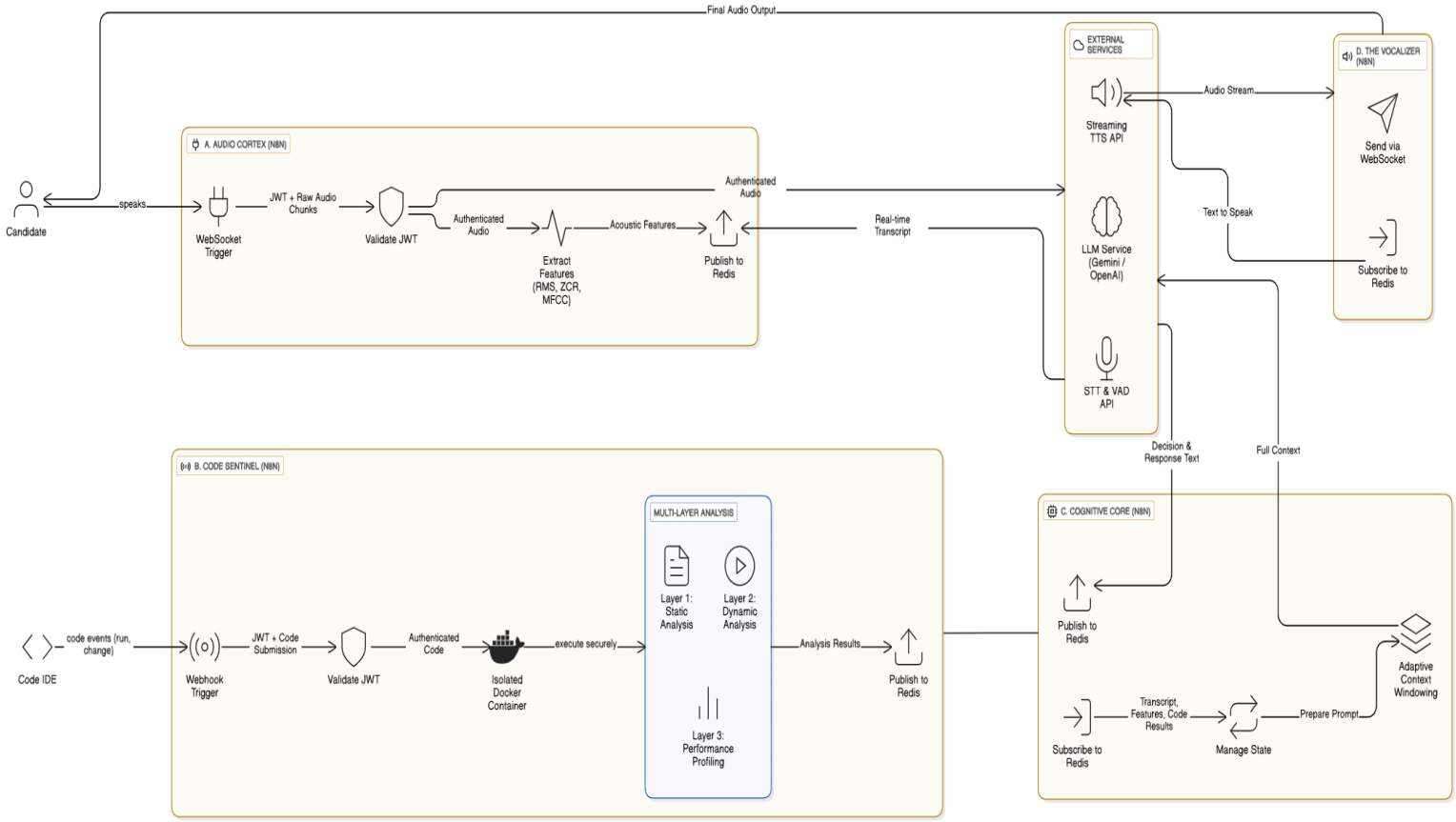
Introduction

To move beyond the limitations of scripted, impersonal interview bots, we set out to build an agent that could listen, understand, and guide like a senior engineer. This document outlines the architecture of this agent, designed to conduct interviews with the nuance and empathy of a human by understanding not just what is said, but how.

Our initial prototypes were too slow, bottlenecked by a single process trying to do everything at once. The solution was to decentralize, creating a multi-workflow architecture named Hydra where specialized agents work in parallel, communicating via a high-speed Redis message broker to achieve seamless, real-time interaction.

- **Core Principle:** Four independent, specialized n8n workflows (Audio, Code, Cognition, Voice) run simultaneously. This event-driven, parallel processing model is the key to minimizing latency.
- **Communication Layer:** A Redis Pub/Sub channel acts as the central nervous system, allowing the workflows to exchange findings and instructions asynchronously and instantly.

Any workflow mentioned here with n8n/Zapier, can be easily integrated using LangChain too. For our hackathon's bonus round we chose the simplest way to implement the flow.



The Hydra Workflows

Each workflow, or "head" of the Hydra, is a master of its domain, contributing a unique piece of the puzzle to the agent's overall awareness.

A. The Audio Cortex: The Master Listener

To truly understand a candidate, the agent must listen beyond the words for cues of confidence or hesitation. The Audio Cortex serves as the dedicated 'ear' of the system, its sole purpose being to process the raw audio stream in real-time and extract these crucial non-verbal signals.

- **Function:** This workflow handles Voice Activity Detection (VAD), streaming transcription (STT), and calculates low-level acoustic features from raw audio chunks.
- **Technical Implementation:** An n8n workflow initiated by a WebSocket Trigger. It uses custom Python scripts for feature calculation and HTTP Request nodes for connecting to streaming STT/VAD services.
- **Key Signals Extracted:**
    - **Hesitation / Low Confidence:** Triggered by a sustained drop in Root Mean Square (RMS) Energy, indicating a drop in vocal amplitude.

      $$RMS_t = \sqrt{\frac{1}{K}\sum_{k=K_{start}}^{K_{end}} S(k)^2}$$

    - **Nervousness / Unvoiced Speech:** Triggered by a significant increase in the Zero-Crossing Rate (ZCR), characteristic of whispering or 's'/'f' sounds.

      $$ZCR = \frac{1}{2(K-1)}\sum_{k=K_{start}+1}^{K_{end}} |\text{sgn}(S(k)) - \text{sgn}(S(k-1))|$$

    - **Emotional Stress / Tone Shift:** Detected via changes in Mel-Frequency Cepstral Coefficients (MFCCs), identified through spectral analysis using a Hann Window.

      $$w(k) = 0.5\left(1 - \cos\left(\frac{2\pi k}{K-1}\right)\right) \quad \text{for } k = 0...K-1$$

B. The Code Sentinel: The Vigilant Pair Programmer

An interview isn't just talk; it's about seeing how a candidate solves problems. The Code Sentinel acts as an over-the-shoulder observer and automated test engineer, providing objective analysis of the candidate's code without disrupting their flow.

- **Function:** Securely executes candidate code, runs a predefined test harness, and performs static analysis.
- **Technical Implementation:** An n8n workflow triggered by a Webhook from the IDE. It uses an SSH or Execute Command node to spin up an isolated, ephemeral Docker container for sandboxed code execution. This is a critical security measure.

C. The Cognitive Core: The Maestro

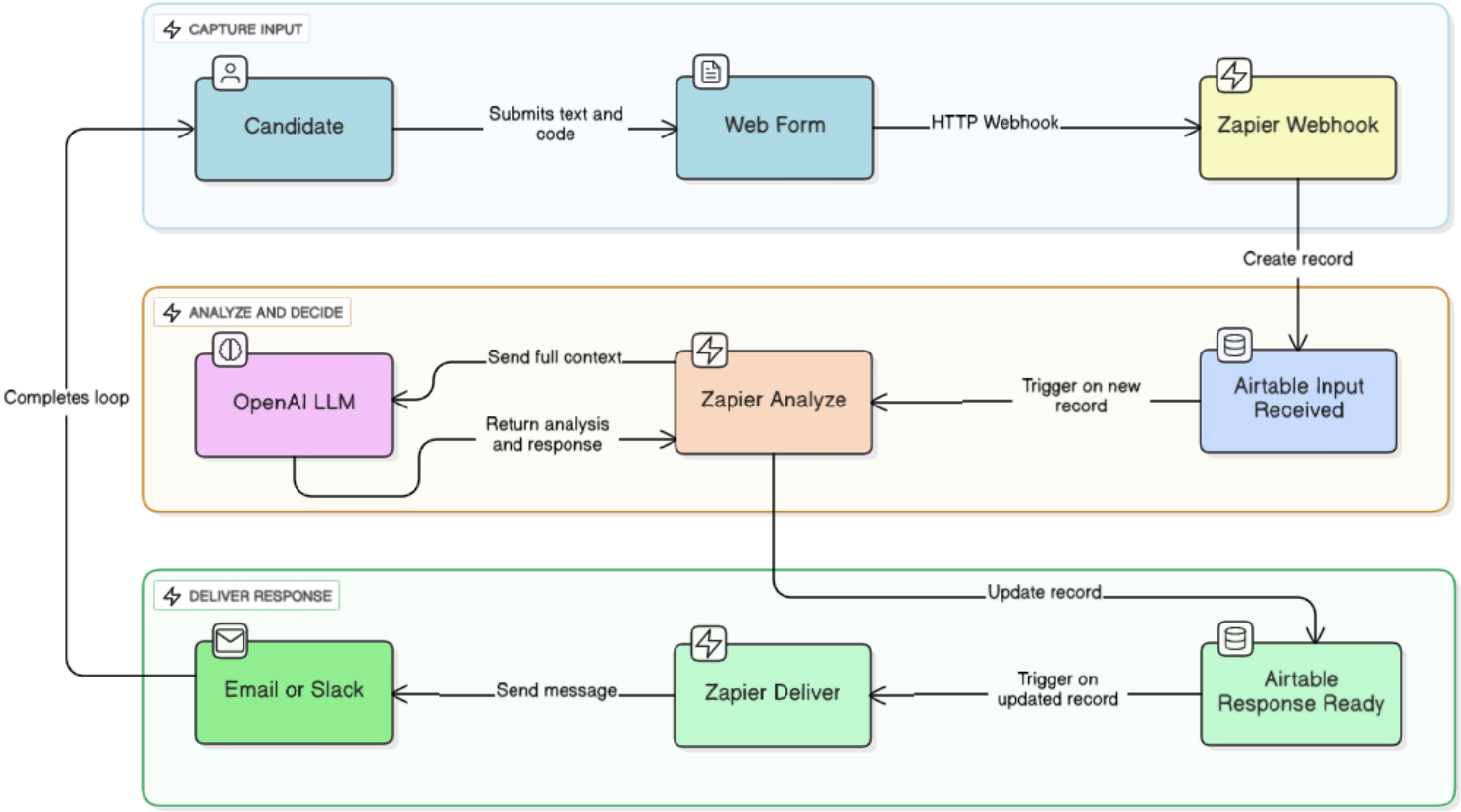
With streams of data coming from the 'ears' and 'eyes', a central mind is needed to make sense of it all. The Cognitive Core is the agent's brain, synthesizing multimodal inputs to make intelligent, context-aware decisions on how to guide the conversation.

- **Function:** Manages the agent's state machine (Listening, Guiding, Probing, etc.). It fuses transcripts, audio analysis, and code results to select the next optimal action using an LLM.
- **Technical Implementation:** An n8n workflow triggered by a Redis Subscribe node. It uses Switch nodes for state management and an LLM Node (e.g., Gemini, OpenAI) for decision-making.

D. The Vocalizer: The Voice

To be truly conversational, the agent must respond as quickly and naturally as a human. The Vocalizer is a simple yet crucial component that transforms the Core's textual decisions into seamless, audible speech.

- **Function:** Converts text responses into a natural-sounding audio stream.
- **Technical Implementation:** An n8n workflow triggered by Redis messages from the Cognitive Core. It uses an HTTP Request node to call a streaming Text-to-Speech (TTS) API like ElevenLabs, pushing audio back to the user via WebSocket.



Deep-Dive: Code Environment Analysis

To provide a truly comprehensive technical assessment, the agent must analyze the candidate's code with the same rigor as a senior engineer. Our Code Sentinel workflow accomplishes this through a multi-layered analysis, moving far beyond simple execution to understand the *quality*, *correctness*, and *efficiency* of the solution.

- **Layer 1: Real-time Static Analysis**

As the candidate types, the Code Sentinel performs lightweight, on-the-fly static analysis. It uses linters (e.g., Pylint, ESLint) and Abstract Syntax Tree (AST) parsing to detect syntax errors, style violations, and anti-patterns in real-time without ever running the code.

- **Layer 2: Dynamic Analysis & Verification**

When the candidate runs their code, the Sentinel executes it within a secure Docker container against a pre-defined test harness. This suite includes unit tests, integration tests, and critical edge cases (e.g., null inputs, empty arrays, large datasets) to objectively verify the solution's correctness and robustness.

- **Layer 3: Performance Profiling**

During execution, the Sentinel also profiles the code's performance, measuring execution time and memory consumption. This provides empirical data that allows the agent to move beyond theoretical Big O notation and discuss the practical performance implications of the candidate's algorithmic choices.

Security: Authentication and Authorization

To ensure the integrity and confidentiality of every interview, the platform is secured using industry-standard protocols. The focus is on ensuring that only the intended candidate can access their specific interview session.

- **JWT-Based Authentication:** The entire session is secured using JSON Web Tokens (JWT). When a candidate logs in, the backend issues a short-lived JWT containing claims like candidate_id and interview_id. This token must be included in all subsequent communication, including the initial WebSocket connection.
- **Authorization Flow:** Our backend services, including the WebSocket server and the n8n workflows, validate the JWT's signature and check its claims before processing any request. This ensures a candidate's data stream is isolated and they cannot access or interfere with any other session.

Project Viability and Challenges

While ambitious, this project is built on a foundation of mature and accessible technologies, making it highly viable. However, it's important to acknowledge and plan for its inherent challenges.

- **Viability**

The architecture's core components—n8n, Docker, and Redis—are open-source, robust, and well-documented. The primary costs are variable API calls for AI services, which can be managed and optimized. The modular, event-driven design means the system is technically feasible and scalable.

- **Shortcomings & Mitigation**
    - **API Costs & Latency:** The reliance on external AI services is the main challenge. This is mitigated by using faster, more cost-effective models (like Groq or Gemini Flash), implementing intelligent Voice Activity Detection to minimize unnecessary transcription, and caching common responses.
    - **Nuance Misinterpretation:** AI can still struggle with deep sarcasm or complex cultural nuances. This is mitigated through continuous prompt engineering and fine-tuning models on domain-specific conversational data.
    - **Orchestration Complexity:** Managing a distributed system is more complex than a monolith. This is addressed by using Docker Compose or Kubernetes to define, deploy, and manage the entire application stack as a single unit.

Conclusion & Unique Value Proposition

This Agentic AI Interviewer is designed not merely to automate a checklist but to fundamentally improve the quality of technical assessments. By creating a responsive, insightful, and adaptive conversational partner, we can gain a far deeper understanding of a candidate's true problem-solving abilities.

The unique value of the Hydra architecture sets it apart from existing solutions:

1. **True Multimodal Fusion:** Its greatest strength is its ability to fuse disparate data streams in real-time. Unlike systems that analyze code *or* speech, Hydra combines low-level acoustic signals (hesitation, tone), linguistic content (transcripts), and multi-layered code analysis into a single, unified context for the LLM. This creates a holistic understanding of the candidate's state.
2. **Inherently Low-Latency Design:** The decoupled, event-driven architecture is our core technical innovation for this use case. While other bots use a slow, sequential request-response cycle, Hydra's parallel workflows enable simultaneous processing, drastically reducing latency and facilitating a natural, free-flowing conversation.
3. **Intelligent, Adaptive Memory:** The Adaptive Contextual Windowing technique is a specific and novel solution to the pervasive latency-vs-context trade-off. It allows the agent to maintain long-term conversational awareness and recall specific past details with precision, making it feel more intelligent and attentive without sacrificing real-time performance.