

Engineering Directive: Hybrid Conversational AI Framework

1. Executive Summary

This document defines the requirements for a high-performance, voice-first AI orchestration framework. The goal is to move beyond "prompt-only" agents by introducing a deterministic logic layer that governs an LLM's behavior, ensuring reliability, sub-1-second latency, and sophisticated voice UX.

2. Core Architectural Principles

2.1 The Sandwich Model

The system must decouple conversational "vibe" from operational "logic."

- **LLM Layer:** Handles natural language understanding (NLU), empathy, and response generation.
- **Logic Engine (Orchestrator):** Enforces business rules, validates data, and manages state.
- **Streaming Layer:** Manages full-duplex audio, Voice Activity Detection (VAD), and text-to-speech (TTS) synthesis.

2.2 Manifest-Based Skill Design

Skills must be modular, testable, and portable. Each skill is defined by a **Skill Manifest** (YAML/JSON):

- **Intents/Triggers:** Natural language patterns that activate the skill.
 - **Required Slots:** Data points the agent *must* collect before executing tools.
 - **Local Toolbox:** A restricted set of APIs available only to that skill.
 - **Dependency Gates:** Prerequisites that must be met (e.g., `is_authenticated == True`).
-

3. Deterministic Operational Logic

3.1 The Logic Compiler & Interceptor

Business rules are defined in natural language by builders but compiled into **Directed Acyclic Graphs (DAGs)** for execution.

- **Rule Enforcement:** If an LLM attempts a tool call that violates a DAG constraint (e.g., "Cannot fetch bills older than 6 months"), the framework must **intercept and block** the call.

- **System Correction:** Upon a blocked action, the framework injects a "System Message" into the LLM's context, forcing it to explain the limitation to the user rather than hallucinating a workaround.

3.2 Python Validation Hooks

To ensure 100% data integrity without LLM overhead:

- Developers can attach **Python scripts** to specific slots.
- These hooks run in **<10ms** to validate formats (Regex), check databases, or transform strings.
- Failure in a hook triggers a "Retry Protocol" where the LLM is instructed to re-ask for the specific missing or invalid data.

4. Voice UX & Latency Management

4.1 Pipeline Fusion

To achieve sub-1-second latency, we must minimize sequential LLM calls.

- **The Single-Turn Call:** The Orchestrator fuses Skill Context, Global State, and User Input into one prompt.
- **Speculative Backchanneling:** A small-latency "Filler Model" fires immediate verbal acknowledgments (e.g., "Let me see...") to mask the processing time of the primary LLM.

4.2 Semantic Interruption (Duck vs. Kill)

The system must distinguish between "Backchannels" (mm-hmm) and "Intent Shifts."

- **Acoustic Track:** Immediate volume ducking (70%) when user audio is detected.
- **Semantic Track:** A real-time classifier analyzes the first 300ms of user speech.
 - *Continuer detected:* Restore volume and keep speaking.
 - *New Intent detected:* Kill audio immediately and pivot logic.

5. State & Handoff Management

5.1 The Shared State Object (SSO)

The SSO is the "Source of Truth" across the session.

- **Persistence:** Tracks user identity, session history, and filled slots.
- **Mapping Layer:** Enables portability. A skill's internal variable `{{target_amount}}` is mapped to the Global State `State.User.Balance` at the deployment level.

5.2 Transition Logic

- **User-Driven:** The Router detects a skill change via intent shift.

- **Business-Driven:** The Logic Engine triggers a "Skill Migration" based on data (e.g., if `balance > 5000`, trigger the `Wealth_Management` skill).

6. Engineering Requirements Table

Feature	Requirement	Implementation Strategy
Latency	< 1.0s (Perceived)	Streaming TTS + Speculative Fillers.
Logic	Deterministic	Interceptor Pattern (Code-based blocking).
Validation	Data Integrity	Sync Python Hooks on Slot-filling.
Scale	Multi-Skill Support	Router-Worker pattern with Shared State.
Voice	Semantic Interruption	Dual-track (Acoustic + Semantic) analysis.

Would you like me to elaborate on the specific data schema for the "Shared State Object" so your engineers have a blueprint for the database layer?