Architecture: Human-in-the-Loop (HITL) Voice Handoff

This setup transitions a live phone call from the automated AI Agent flow to a human agent dashboard built in **Angular**, using **WebSockets** for signaling and **WebRTC** for real-time audio transmission.

1. Triggering the Handoff

The process begins within the existing backend bot flow. When the logic engine hits the **Agent Bot Node**:

- The backend pauses the AI's response generation.
- The server identifies the active session and emits an incoming_call event via the **Socket connection** to the Angular client.
- This payload should include the threadId (for **alive5** syncing) and caller metadata.

2. Signaling & WebRTC Establishment

The Angular application acts as the signaling endpoint.

- **User Interaction:** Upon receiving the socket event, the Angular app displays a call notification. The human agent clicks "Accept."
- **The Handshake:** The Angular app initiates the WebRTC handshake (Offer/Answer and ICE candidates) through the existing Socket connection.
- **Media Stream:** Once the PeerConnection is established, the browser's microphone audio is sent as a stream to the server.

3. Audio Routing & Stream Replacement

This is the most critical part of the backend implementation. The server must act as a media bridge between the Telephony provider (e.g., Twilio/Vonage) and the WebRTC stream.

- **Replacing the AI:** The backend stops piping the AI's **Text-to-Speech (TTS)** stream to the phone line.
- **Direct Bridging:** The server routes the raw audio coming from the Angular WebRTC stream directly into the phone call. Conversely, the caller's audio is routed back to the Angular app.
- **Full Duplex:** This ensures a low-latency, natural conversation between the human and the caller.

4. Alive5 Thread Persistence (Shadow Transcription)

To ensure the conversation history remains intact within **alive5**, the human-led portion of the call must still be digitized.

- **Real-time STT:** While the server is bridging the human's audio to the phone, it must simultaneously run that audio through a **Speech-to-Text (STT)** engine.
- **Socket Updates:** The resulting text is then pushed to the **alive5 backend** using the existing socket events.
- **Outcome:** The **alive5 thread** will show a seamless transition from "Bot responses" to "Human responses" in the chat log, even though the interaction was via live voice.