frontdesk.

Frontdesk Engineering Test: Human-in-the-Loop AI Supervisor

Context

At Frontdesk, we're building AI receptionists that can actually manage customer relationships end-to-end.
Today, if our AI doesn't know something, it either hallucinates or fails. That's not good enough. We want it to behave like a real agent — escalate intelligently, learn, and get smarter over time.

This project is the foundation for that. It will also be your first exposure to how we think about code quality, modularity, and product UX.

This project is **due one week from receipt**.

Goal

Build the first version of a **human-in-the-loop system** for our AI agents.
If the AI doesn't know the answer, it should **escalate to a human supervisor**, **follow up with the original customer**, and **update its knowledge base automatically**.

Deliverables

You will build a small, locally running system that can receive phone calls and loop in a human when necessary.

All tasks below can be completed using free credits.

---

1. AI Agent Setup (First Step)

- Set up a **basic simulated AI agent** using LiveKit (you can use their Python SDK or another language you prefer).
    - *Why LiveKit? We are evaluating your ability to research an unfamiliar framework, read documentation, and debug in a foreign environment.*

- Prompt it with basic business information about a **fake salon**.
    - *Why LiveKit? We want to evaluate your ability to build good prompts.*

- The agent doesn't need full conversational handling. You can just use a sample app and wrap up this step quickly. You just need to equip your agent with the ability to:

    - Receive a call

    - Respond if it knows the answer

    - Trigger a "request help" event if it doesn't know

---

2. Human Request Handling

When the AI doesn't know something:

- Tell the caller: **"Let me check with my supervisor and get back to you."**

- Create a **pending help request**.

    - You decide how this request is structured in a DB— we will evaluate the elegance of your solution. You can use a lightweight DB like DynamoDB/Firebase to move fast.

- Simulate texting the supervisor:

    - E.g., log a console message, or trigger a webhook.

- Content: “Hey, I need help answering [question].”

---

3. Supervisor Response Handling

- Build a simple UI where a human can:
  - View pending requests
  - Submit answers to pending requests
  - See history of resolved/unresolved requests
- When the supervisor responds:
  - AI should **immediately text back** the original caller (simulated via webhook or console log).
  - AI should **update its internal knowledge base** with the new information.

---

4. Knowledge Base Updates

- Implement a basic system where **learned answers are saved**.
- Include a simple view like a “Learned Answers” section.

Things You Must Handle

- Requests must have a **lifecycle**: `Pending` → `Resolved` / `Unresolved` (on timeout).
- Supervisor responses must **link cleanly** back to the originating request and customer.
- The AI must **follow up immediately** once a supervisor responds.

Design Decisions (You Own These)

We want you to make clean design choices — we'll review these closely:

- How you model "help requests" (tables, fields, relationships)
- How you structure updates to the knowledge base
- How you handle supervisor timeouts gracefully
- How you think about scaling from 10/day to 1,000/day
- How you modularize agent, help requests, and text-back handling

---

Constraints

- Keep the UI extremely simple — think internal admin panel, not polished product.
- You may simulate calls/texts using console logs or webhook calls. (No need to integrate Twilio unless you want to.)
- Prioritize **code quality** and **architecture** over completeness.
- Assume this service must **run reliably without human babysitting** — handle basic errors.

Submission Instructions

By the end of the week:

- Send us a **GitHub repository** with your code and a simple README (setup + design notes).
- Record and submit a **short video demo** (screen recording is fine) walking us through:
    - How your system works
    - Any key decisions you made
    - What you would improve next

This doesn't need to be highly polished — we care most about clear thinking and clean code.

Why We're Doing It This Way

We are deliberately leaving some parts a little open-ended.
We want to see:

- How you reason through ambiguous product specs
- How you balance speed versus structure
- How you design systems that **improve themselves over time**

Phase 2 (For Later Discussion)

If the supervisor is available during a live call, the AI should offer to **put the caller on hold**, **transfer the call**, and **resolve live**.
If not, it should fall back to the text-based flow you built in Phase 1.

We'll discuss how you would approach this after you submit Phase 1.

Timeline

You will have **one week** from receiving this assignment.

We expect this to take around **10–15 focused hours** spread out over the week, depending on how much polish and completeness you add.
It doesn't need to be perfect — we care most about structure, clarity, and solid judgment.

Next Steps

We are currently hiring across multiple engineering roles.
If your submission shows strong thinking and execution, you will likely be invited to a **final-round interview** to walk us through your project, discuss your design decisions, and potentially start a **trial period** with us to join full-time.

Thanks

We're excited to see how you think and build.
Good luck!