As part of this project, we will be inquiring how you used AI tools to complete this assignment. We understand this project sounds very complicated, but if you are using AI coding tools, this should be very easy to complete. This is a less complex single use version of a tool we have live deployed!

If you are not based in the USA, please use web calls to test the application instead of phone calls.

Take-Home Project: Build an AI Voice Agent Tool

Objective:

Your task is to build a functional web application that allows a **non-technical administrator** to configure, test, and review calls made by an adaptive AI voice agent. The project centers on creating a simple and intuitive UI for three core administrative functions: **configuring** the agent's logic, **triggering** test calls, and **analyzing** the structured results.

Recommendations:

You can and should use AI for coding this assignment

You have 4 days to complete

Make sure to have a consistent and clear commit history, make sure to do a clear and trackable **START** commit, and an **END** commit; both should be easily identified.

Code quality matters. As a Senior developer, you will be required to know good modularization practices, keeping the patterns that you have established, consistency of decisions and other topics will be evaluated.
EG: It's also important to know when speed matters much more than a good piece of code that is used only in one place

If all the requirements were accomplished, you can feel free to add more things at your will to make it better, add some nice features, or others. Make sure to mention those in the README section

Part 1: Core Requirements & Technology Stack

You are to build a web application using **React**, **Typescript, FastAPI**, and **Supabase**. You will use **Retell AI** for the voice system. The application you build must meet the following requirements for its administrative user:

1. **Agent Configuration UI:** The application must provide a simple UI that allows an administrator to define the prompts and logic that guide the agent's conversations.
2. **Call Triggering & Results UI:** From the dashboard, the administrator must be able to:
    - Enter the driver's name, phone number, and the relevant load number into fields to provide context for the call.

    - Click a "Start Test Call" button to trigger a phone call from the configured voice agent.
    - After the call is complete, view the results in a structured, easy-to-read format. This summary should present the key information collected during the call as clear key-value pairs, alongside the full call transcript.
3. **Backend Logic:** Your FastAPI backend will serve as the webhook for Retell AI, containing the logic to interpret the prompts from the database and guide the agent's conversation in real-time. Your backend must also include a post-processing step to structure the raw transcript into the clean summary displayed in the UI.

Part 2: Project Task - Implement and Test Logistics Agents

The web application you build will be used to implement and test the two logistics agent scenarios detailed below. Your submission should demonstrate that your platform can successfully configure and run these agents.

Task A: Implement Optimal Voice Configuration

Your agent configurations should demonstrate best practices for a realistic voice experience. In your implementation, you must make use of Retell AI's advanced settings, such as **backchanneling**, **filler words**, and **interruption sensitivity**, to make the agent sound as human-like as possible.

Scenario 1: Logistics - End-to-End Driver Check-in ("Dispatch")

- **Context:** The agent is calling a driver about a specific load (e.g., Mike, Load #7891-B from Barstow to Phoenix). The system knows the load's details but does not know the driver's current status (they could be mid-transit or have just arrived).
- **Goal:** Configure the agent to handle the entire check-in conversation as a single, fluid thread. The agent must first determine the driver's status by asking an open-ended question like, "Hi Mike, this is Dispatch with a check call on load 7891-B. Can you give me an update on your status?" Based on the driver's response, the agent must dynamically pivot its line of questioning.
- **Structured Data to Collect (Success Case):** Your system's post-processing must extract and display the following structured data for the administrator:
    - call_outcome: "In-Transit Update" OR "Arrival Confirmation"
    - driver_status: "Driving" OR "Delayed" OR "Arrived" OR "Unloading"
    - current_location: (e.g., "I-10 near Indio, CA")
    - eta: (e.g., "Tomorrow, 8:00 AM")
    - delay_reason: (e.g., "Heavy Traffic", "Weather", "None")
    - unloading_status: (e.g., "In Door 42", "Waiting for Lumper", "Detention", "N/A")
    - pod_reminder_acknowledged: true OR false

Scenario 2: Logistics - Dynamic Emergency Protocol ("Dispatch")

- **Context:** The agent is in the middle of a routine check call when the driver interrupts with

an emergency (e.g., "I just had a blowout, I'm pulling over").
- **Goal:** This is the most critical test. Configure your system so the agent can **immediately abandon its standard conversation thread** in response to an emergency trigger phrase. It must gather critical information and then escalate by stating it is connecting the driver to a human dispatcher.
- **Structured Data to Collect (Success Case):** In an emergency, the structured summary must contain:
  - call_outcome: "Emergency Escalation"
  - emergency_type: "Accident" OR "Breakdown" OR "Medical" OR "Other"
  - safety_status: (e.g., "Driver confirmed everyone is safe")
  - injury_status: (e.g., "No injuries reported")
  - emergency_location: (e.g., "I-15 North, Mile Marker 123")
  - load_secure: true OR false
  - escalation_status: "Connected to Human Dispatcher"

Task B: Implement Dynamic Response Handling

Your implementation must also gracefully handle the following special cases, which will be used to test your system's robustness:

- **The Uncooperative Driver:** The agent should be able to probe for more information when given one-word answers and know when to end the call if the driver remains unresponsive.
- **The Noisy Environment:** The agent should be able to handle garbled speech-to-text results by asking the driver to repeat themselves a limited number of times before escalating.
- **The Conflicting Driver:** The agent should be able to handle discrepancies between the driver's stated location and the system's GPS data in a non-confrontational way.

Deliverables

1. **A link to a Git repository** containing your complete, functional web application.
2. **(Optional but Recommended)** A short, unlisted video (e.g., Loom) demonstrating your application in action. Show how an administrator would configure an agent, trigger a call, and view the results for one of the scenarios.
3. A brief README.md in your repository explaining your design choices and how to run the application.