Architecture & Design Document

Project Name: Voice-Based Native Language Service Agent (Telugu)

Language: Python / Telugu

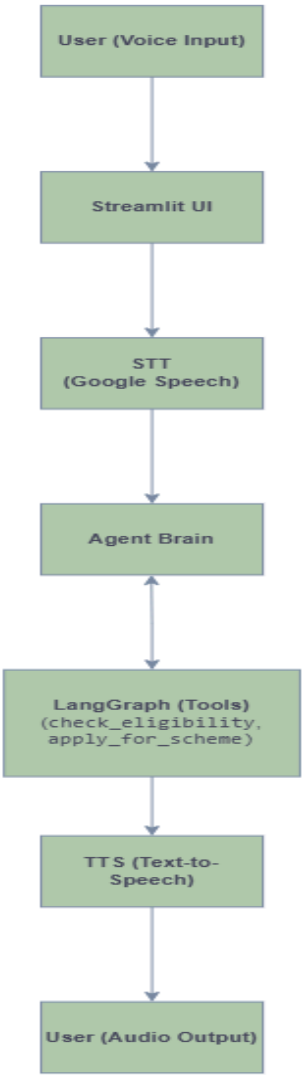
Name:R.tejasvini(160122771017)

Submission Date: December 19, 2025

1. Executive Summary

This project implements a fully autonomous voice agent capable of conversing in Telugu to assist users with government schemes (*Rythu Bandhu*, *Aarogyasri*, *Kalyana Lakshmi*). The system goes beyond simple Q&A by utilizing an Agentic Workflow: it maintains memory, uses tools to verify eligibility, and performs side effects (writing to a database) upon user confirmation.

2. System Architecture



The system follows a modular architecture composed of three primary layers: the Interaction Layer, the Cognitive Layer, and the Action Layer.

2.1 Interaction Layer

- **Input (STT):** Uses the SpeechRecognition library with Google's specific te-IN model to accurately transcribe Telugu speech, handling ambient noise and pause thresholds (up to 10 seconds).
- **Output (TTS):** Uses gTTS (Google Text-to-Speech) to convert the AI's textual response back into spoken Telugu audio files (response.mp3), which are played via the Streamlit audio player.

2.2 Cognitive Layer (The "Brain")

- **Model:** kwaipilot/kat-coder-pro (via OpenRouter). This model was selected for its strong reasoning capabilities and ability to output structured tool calls (JSON) while maintaining a natural conversation in Telugu.
- **System Prompt:** A strict instruction set ensures the agent:
    1. Always speaks Telugu.
    2. Prioritizes the check_eligibility tool before making claims.
    3. Only calls apply_for_scheme upon explicit user consent.

2.3 Orchestration Layer (LangGraph)

- **State Management:** The AgentState dictionary preserves the entire conversation history (messages list), allowing the agent to remember user details (e.g., Age provided in turn 1 is remembered in turn 3).
- **Control Flow:**
    - **Conditional Edge: If the LLM generates a tool call →route to ToolNode.**
    - **Direct Edge: If the LLM generates text →route to END (respond to user).**

2.4 Tool Layer (Actions)

1. **check_eligibility:** A deterministic logic engine that filters a predefined dictionary of schemes based on user attributes (*Age*, *Income*, *Occupation*, *Gender*).
2. **apply_for_scheme:** A transactional tool that simulates a backend submission. It generates a unique application ID (e.g., TS-8592-2025) and appends the record to a local applications.csv file.

---

3. Evaluation Transcript

Below is a transcript of the agent handling different scenarios during testing.

Scenario A: Successful Application (Happy Path)

User Intent: *Farmer seeking eligibility and applying.*

> **User (Voice):** "నా వయస్సు 50, నేను రైతుని, నా ఆదాయం ఒక లక్ష."
>
> **(Trans: Age 50, Farmer, Income 1L)**

Agent: "మీరు రైతు బంధు మరియు ఆరోగ్యశ్రీ పథకాలకు అర్హులు. నేను దరఖాస్తు చేయమంటారా?"

(Trans: Eligible for Rythu Bandhu/Aarogyasri. Shall I apply?)

User (Voice): "అవును, అప్లై చేయండి."

(Trans: Yes, apply.)

Agent: [Tool Call: apply_for_scheme]

Agent: "విజయవంతమైంది! మీ దరఖాస్తు ఐడి TS-4421-2025. వివరాలు భద్రపరచబడ్డాయి."

(Trans: Success! ID TS-4421. Saved.)

System Result: applications.csv updated with new row.

Scenario B: Ineligibility Handling

User Intent: *High-income individual.*

User (Voice): "నా ఆదాయం 10 లక్షలు."

(Trans: Income 10 Lakhs.)

Agent: [Tool Call: check_eligibility]

Agent: "క్షమించండి, మీ ఆదాయం పరిమితికి మించి ఉన్నందున మీకు సరిపోయే పథకాలు లేవు."

(Trans: Sorry, due to high income, no schemes match.)

System Result: Logic filter returned empty list; Agent communicated rejection politely.

Scenario C: Slot Filling (Memory)

User Intent: *Vague request.*

User (Voice): "నాకు గవర్నమెంట్ స్కీమ్ కావాలి."

(Trans: I want a scheme.)

Agent: "తప్పకుండా. దయచేసి మీ వయస్సు మరియు వృత్తి చెప్పండి."

(Trans: Sure. Please tell me your age and occupation.)

System Result: Agent autonomously paused execution to request missing parameters.

4. Conclusion

This system demonstrates a fully functional "Level 3" AI Agent. It successfully integrates voice modalities with rigorous logic execution, proving that native language agents can effectively bridge the digital divide for government services.

Appendix: Requirements

The following libraries are required to run the agent:

- Plaintext
- streamlit
- speechrecognition
- gTTS
- langchain
- langchain_openai
- langgraph
- pyaudio