AI Call Assistant

Intelligent Call Management

Proof of Concept for Automated Call Handling using Conversational AI

Hiya Assessment - By Kumuda Aggarwal
MS in Computer Science, New York University

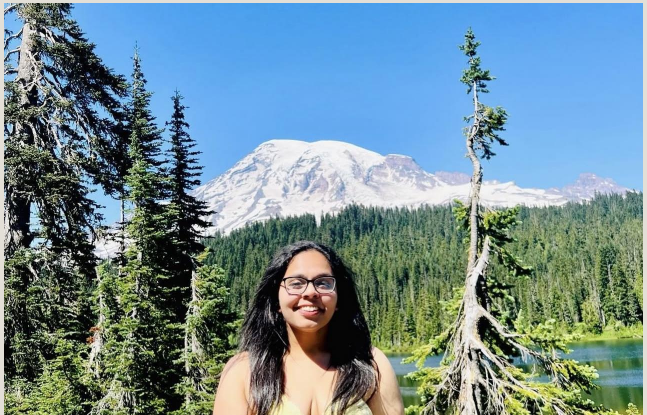
About me

M.S. in Computer Science, New York University
B.S. in Computer Science, University of Texas at Dallas

💼 **Amazon | Natera | Splunk (Cisco)**

🌎 **Fun Fact:** Lived in Seattle 3 years -love Olympic NP Rainier & Cascades. Summer is my favorite!

🎬 **Favorite Shows:** The Office, Schitt's Creek, Suits....



Customer Scenario & Rationale

Scenario (Jobs-to-Be-Done Format):

> *"As a smartphone user, I want an AI agent to manage my incoming calls and let me interact via voice commands so that I can handle important calls, avoid spam, and route calls appropriately without touching my phone.*

Problem Context:

- Users receive a mix of **important, spam, telemarketing, and unknown calls** daily.
- Current solutions need manual effort and make handling calls while busy or driving unsafe..

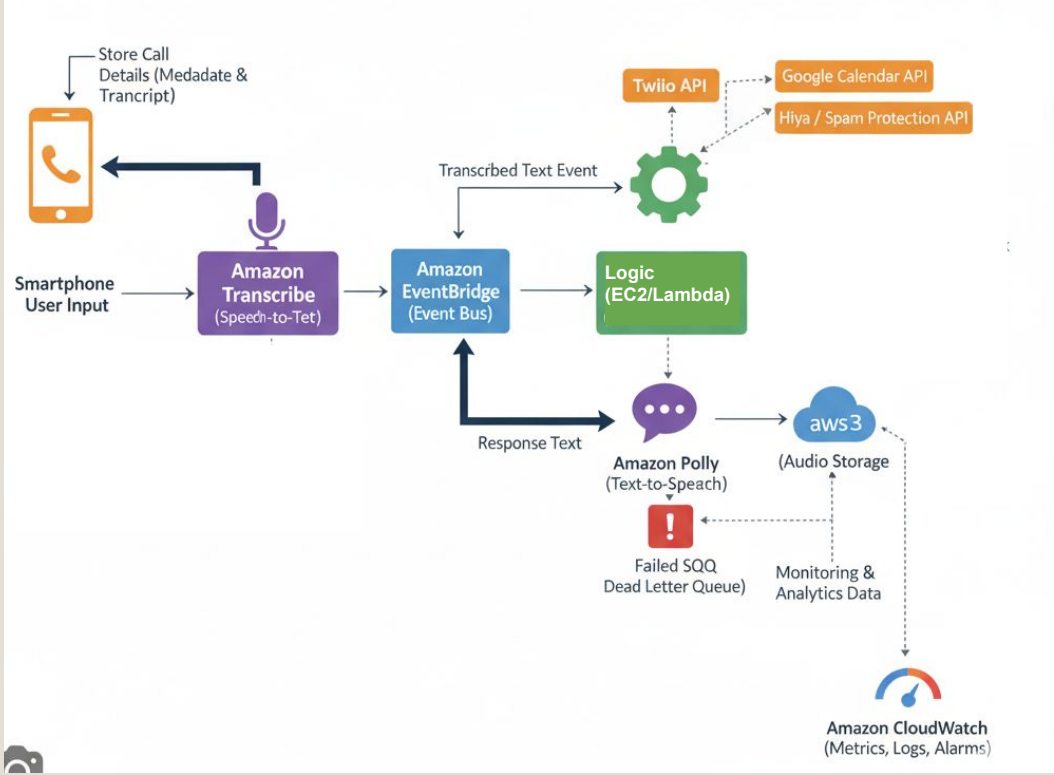
AI Solution:

- Handles voice or text commands to pick, block, voicemail, forward, or schedule callbacks 📞
- API calls simulate integration with **Twilio and Hiya APIs**.

Benefit to Users:

- **Save time** by quickly filtering spam calls.
- **Reduce frustration** with unwanted interruptions.
- **Accessible & safe** — usable while driving or multitasking.

Technical architecture



Live Demo

Github - https://github.com/Kumuda123/Hiya-AI-Call-Assistant

Technology Stack and Orchestration Framework

Component	Current (POC)	Future/Potential
Command Intent / NLP	Java-based custom parser	Advanced NLP / Amazon Lex / ML intent recognition
Database / Storage	In-memory / local storage	AWS DynamoDB / RDS for persistent call history
Telephony Integration	Twilio Lookup API (mocked)	Live Twilio & Hiya APIs for real call routing, blocking, and voicemail
Speech (STT & TTS)	N/A	AWS Transcribe (Speech-to-Text), Amazon Polly (Text-to-Speech)
Orchestration / Architecture	Single-threaded console loop	Microservices / Event-driven architecture on AWS (Lambda, ECS, S3)

Next Steps

1. **Google Calendar Integration:** Automatically schedule callbacks and check user availability in real time.
2. **Voice Integration**: Connect to speech recognition APIs (e.g., Google Speech, AWS Transcribe) for a true hands-free experience.
3. **Real Telephony Integration**: Integrate live Twilio or Hiya APIs for actual call routing, blocking, and voicemail handling.
4. **Dashboard**: Build a Spring Boot dashboard to display call history, blocked numbers, scheduled callbacks
5. **Unit Testing Suite**: Implement JUnit tests for all core modules.
6. Refactor logic for scalability and modularity by:
   a. Using Natural Language Processing (NLP) for intent recognition.
   b. Adopting a microservices-based architecture for separation of concerns.
   c. Deploying components on AWS for cloud scalability, fault tolerance, and performance.

🎉 **Outcome:** A cloud-hosted, AI-powered call assistant capable of real-time call handling, intelligent routing, and seamless user experience.

Thank you :)