growth loops

Growth Loops Technology Pvt Ltd

Engineering Portfolio · Case Study · Full-Stack Engineering

WingManAI

AI Conversation Intelligence for Professional Networking

ROLE	DOMAIN	STACK
Full-Stack Engineer	**Professional Networking · AI**	**NestJS · Python · GPT-4o**

2	5+	3	19+
Backend Services	AI Pipeline Stages	Subscription Tiers	Analysis Dimensions

About this Project

WingManAI is a production-ready AI platform that captures professional conversations with mutual consent, transcribes them with speaker diarisation, and delivers 19-dimension analysis spanning language, acoustics, and relational dynamics. Growth Loops Technology designed and built both backend services — a NestJS application server and a Python AI microservice — from initial architecture to production deployment.

Document Contents

gunendu@growth-loop.io · +91 9880806633 · Cabin No. 23, Mani Casadona, Kolkata 700161

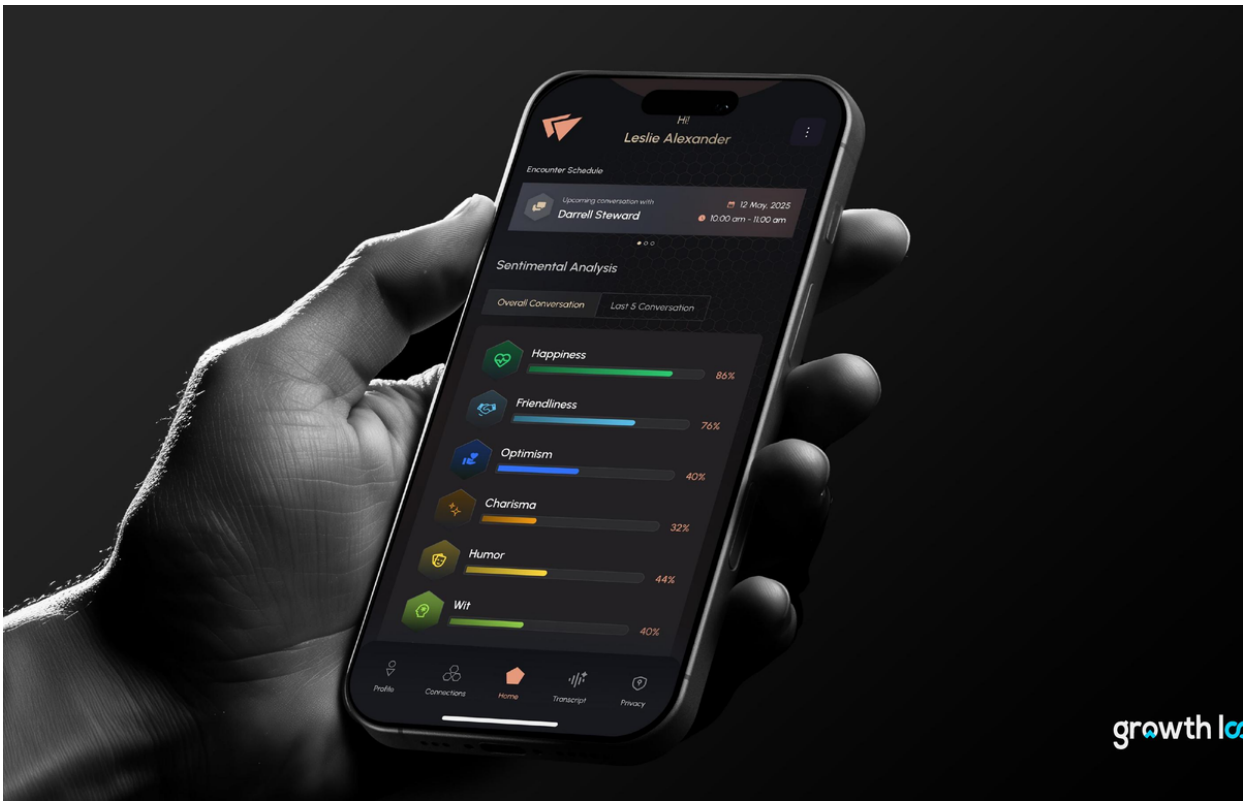
01 — OVERVIEW

What is WingManAI?

WingManAI is an AI-powered professional networking platform designed to make in-person meetings more intentional and analytically insightful. Users connect with professionals, schedule proximity-aware meetings, record their conversations (with mutual consent), and receive deep AI-driven analysis of their communication dynamics, speaking styles, and conversation outcomes.

The platform addresses a universal gap in professional networking: meetings happen, but almost nothing actionable comes out of them. WingManAI captures the full richness of a real conversation, transcribes it, analyzes it acoustically and semantically, and surfaces structured insights that help professionals improve how they communicate and connect.

2	5+	3	19+
Backend Services	AI Pipeline Stages	Subscription Tiers	Analysis Dimensions



WingManAI home dashboard — sentiment analysis scores and upcoming encounters

02 — THE PROBLEM

Professional meetings are a black box

Despite being a cornerstone of professional growth, in-person conversations leave almost no structured record. Professionals walk out of networking meetings with vague impressions and little to act on. Existing tools either record passively or analyze text after the fact — but none connect the acoustic, linguistic, and relational dimensions of a conversation into a coherent, actionable picture.

The core challenges were threefold:

Consent & Privacy: Recording professional conversations requires explicit, granular consent management. Users need control over who can access their audio, transcript, or both, and for how long.

Pipeline Complexity: **Turning raw audio into structured insights requires multiple asynchronous stages — each with different failure modes, latency profiles, and compute requirements.**

Analysis Depth: **Surface-level transcription wasn't enough. The product required understanding not just what was said, but how — pitch, volume, pacing, emotional tone, persuasion patterns, and conversational dynamics.**

03 — ROLE & RESPONSIBILITIES

Full-Stack Engineer — Growth Loops Technology Pvt Ltd

Growth Loops Technology was responsible for the full design and implementation of both backend systems — the NestJS application server and the Python AI microservice — from initial architecture through production deployment. This included:

- Designing the full data model across 25+ Prisma-managed PostgreSQL tables
- Building the NestJS REST API with real-time WebSocket support
- Architecting the async job processing pipeline using BullMQ and Redis
- Building the Python FastAPI AI microservice and integrating it with the main backend
- Designing and implementing the two-tier LLM analysis system using LangChain and GPT-4o
- Integrating AssemblyAI for speaker-diarized transcription
- Implementing the acoustic analysis pipeline using librosa and SpeechBrain
- Integrating Stripe for subscription billing across three tiers
- Setting up CI/CD pipelines with GitHub Actions and Docker

04 — ARCHITECTURE

Two services, one seamless experience

WingManAI's backend is composed of two purpose-built services that work in concert. The separation was a deliberate architectural decision: the compute profile of the AI workload (GPU-intensive, long-running) is fundamentally different from the API workload (low-latency, high-concurrency), and keeping them decoupled allowed each to scale independently.

Service 1 — NestJS Application Server

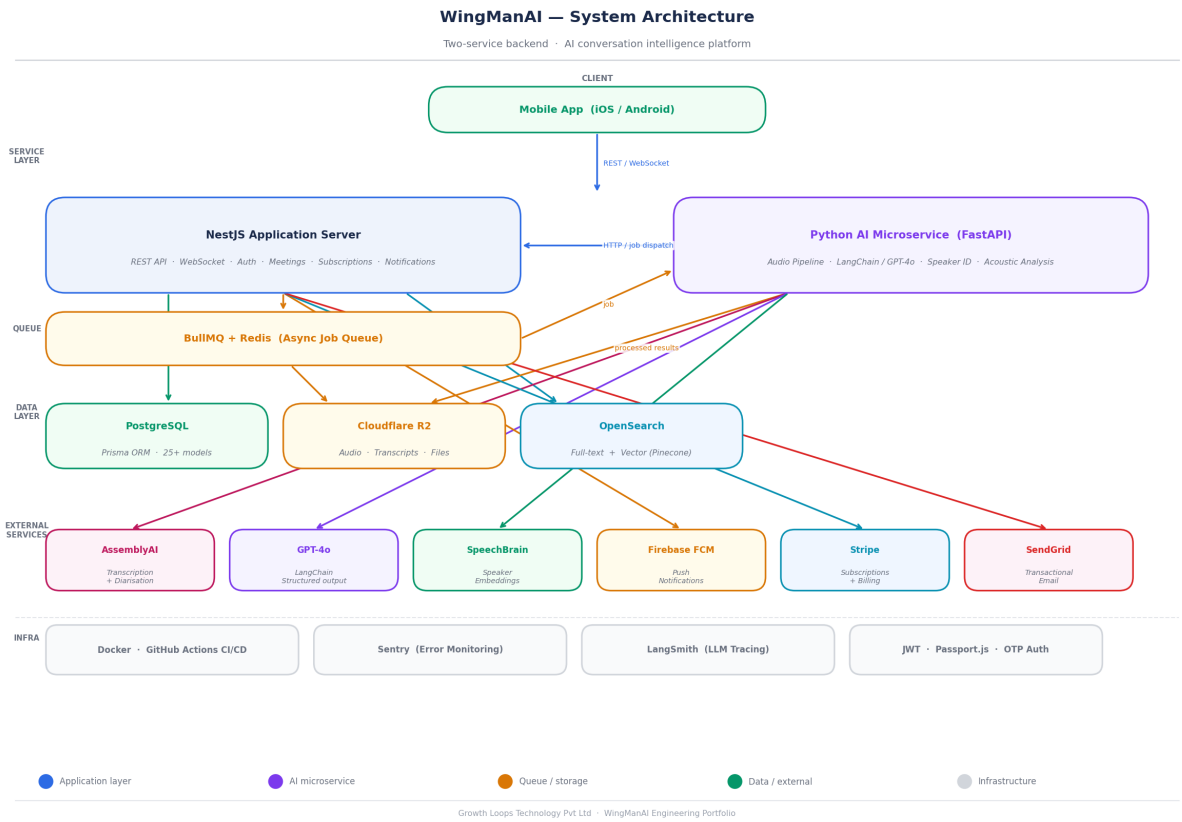
The primary backend handles all product logic: authentication, user management, connections, meeting scheduling, notifications, subscriptions, and real-time communication. It exposes a REST API backed by PostgreSQL (via Prisma ORM), with Redis for caching and BullMQ for async job orchestration.

Service 2 — Python AI Microservice (FastAPI)

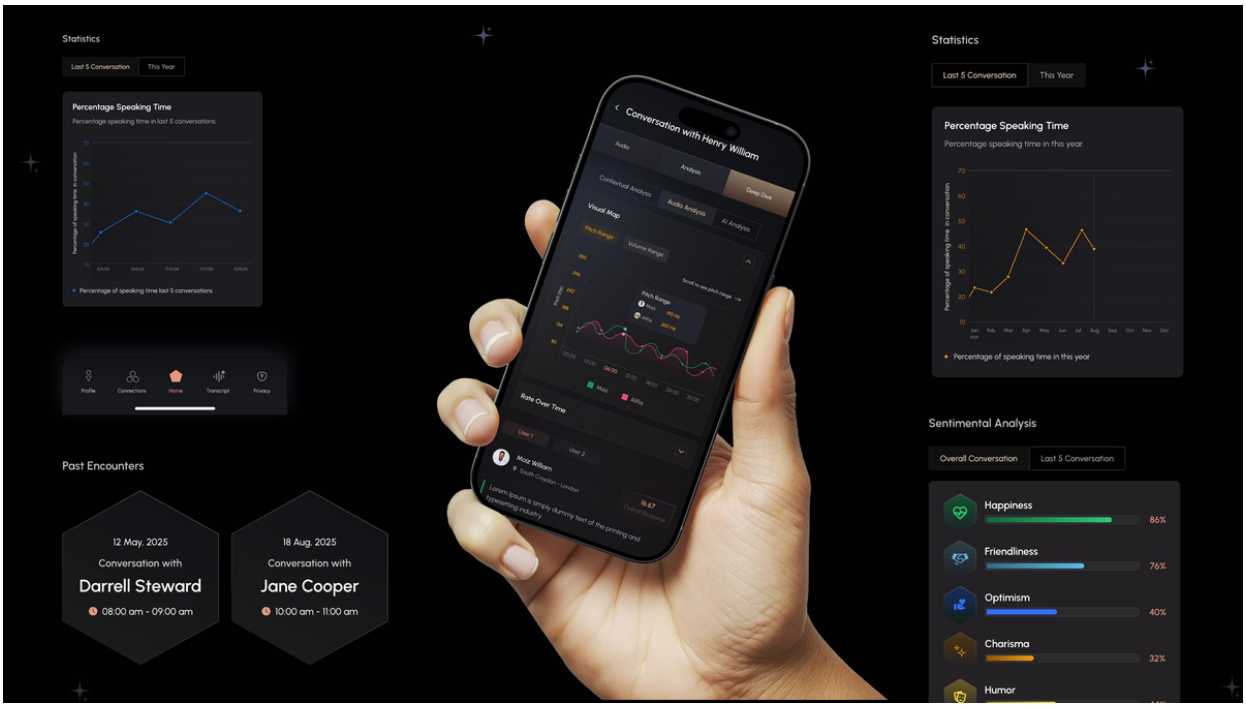
A standalone FastAPI service responsible for all AI and signal-processing workloads. It receives jobs from the main backend, runs the full processing pipeline, and returns structured JSON results. Kept stateless and containerized for easy horizontal scaling.

End-to-End Flow

Mobile App → NestJS API → BullMQ Queue → Python AI Service → PostgreSQL / R2 Storage → Firebase Push



System architecture — end-to-end data flow across both services, queue, data layer, and external integrations



Conversation analysis screen — pitch range, volume visualisation, and per-speaker metrics

05 — KEY ENGINEERING CHALLENGES

Where the hard problems lived

1. Consent-First Audio Recording

Recording professional conversations is sensitive. A dual-consent system was built — each participant independently grants or revokes access to their audio, their transcript, or full combined access, with time-bounded consent windows. A separate consent model allows users to request transcript redaction before sharing data with any third-party professional (Tier 3). Consent state is tracked in its own normalized schema and checked before any downstream processing begins.

2. Async Processing Pipeline

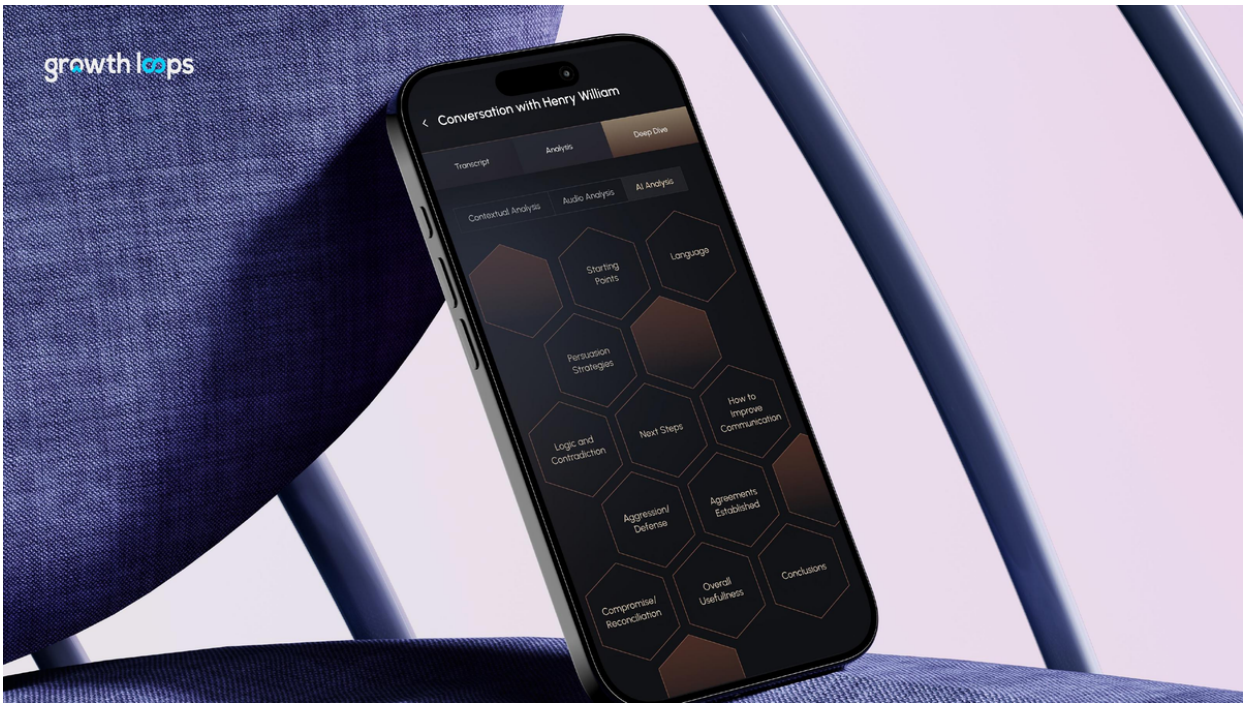
After a meeting ends, multiple long-running tasks run in a defined order: audio merging, preprocessing, transcription, Tier 1 analysis, Tier 2 analysis, audio intelligence, and notification delivery. Built as a multi-stage BullMQ queue pipeline with retry logic, job status tracking, and database state synchronization so the frontend can reflect real-time progress.

1	2	3	4	5	6
Raw Audio	Preprocess	Transcribe	Tier 1 LLM	Tier 2 LLM	Audio Intel

3. Two-Tier LLM Analysis

Each conversation is analyzed in two sequential LLM tiers powered by GPT-4o via LangChain with structured Pydantic output. Tier 1 produces an initial analysis of each participant and the overall conversation. Tier 2 goes deeper — persuasion strategies, conflict areas, agreements reached, and future communication strategies. Both tiers run per-user and conversation-level analysis in parallel using asyncio.

Analysis Layer	Dimensions Covered
Tier 1 (per user)	Speaking style · Key themes · Starting points · Language patterns · Communication approach
Tier 1 (conversation)	Conversation flow · Shared topics · Interaction dynamics · Conversational logic
Tier 2 (per user)	Persuasion strategies · Aggression/defensiveness · Wit · Future communication steps · Constructive vs. accusatory language
Tier 2 (conversation)	Agreements established · Compromises · Overall effectiveness · Conclusions · AI next steps
Audio Intelligence	Pitch peak/min/avg · Volume peak/min/avg · Per-word acoustic profiling · Z-score normalization · Conversation flow chart



AI deep dive — contextual analysis with hexagon topic map across persuasion, logic, and agreement dimensions

4. Acoustic Intelligence

A word-level acoustic analysis pipeline was built using librosa. For each conversation, the system loads the merged audio file, computes STFT, extracts pitch via piptrack, calculates RMS volume per frame, and maps these values onto each sentence and word from the diarized transcript. Per-speaker statistics are computed and normalized to z-scores for cross-conversation comparability. This acoustic data is then fed into an LLM that generates a flowchart of the conversation's dynamics.

5. Speaker Recognition & Voice Profiling

SpeechBrain's speaker embedding model was integrated so that each user's voice can be fingerprinted from a short enrollment sample. When a conversation is transcribed, speaker embeddings are matched against stored voice profiles using cosine similarity — mapping AssemblyAI's generic labels to real user identities.

6. Subscription & Monetisation

Three subscription tiers are integrated with Stripe. Tier 1 provides basic conversation analysis. Tier 2 unlocks deeper thematic analysis and audio intelligence. Tier 3 adds a professional consultation feature — users can share conversation data (with scoped consent) with a credentialed professional via a private real-time chat interface. The full

Stripe webhook pipeline, lifecycle management, invoice tracking, and payment method handling were built from scratch.

06 — TECH STACK

Technology decisions

Layer	Technology
Application Server	NestJS (TypeScript) · REST API · WebSockets (Socket.io)
AI Microservice	Python · FastAPI · LangChain · GPT-4o
Database	PostgreSQL · Prisma ORM (25+ models, 30+ migrations)
Job Queue	BullMQ · Redis
Audio Processing	librosa · noisereduce · pyloudnorm · pydub · SpeechBrain · ffmpeg
Transcription	AssemblyAI (speaker diarization)
LLM Framework	LangChain · Pydantic structured output · LangSmith tracing
Storage	Cloudflare R2 (audio · transcripts · processed files)
Search & Discovery	OpenSearch (Elasticsearch-compatible) · Pinecone (vector search)
Notifications	Firebase Cloud Messaging · SendGrid (email)
Billing	Stripe (subscriptions · webhooks · invoices · payment methods)
Auth	JWT · Passport.js · OTP verification · bcrypt
Infrastructure	Docker · GitHub Actions CI/CD · Sentry (error monitoring)

07 — OUTCOME & REFLECTIONS

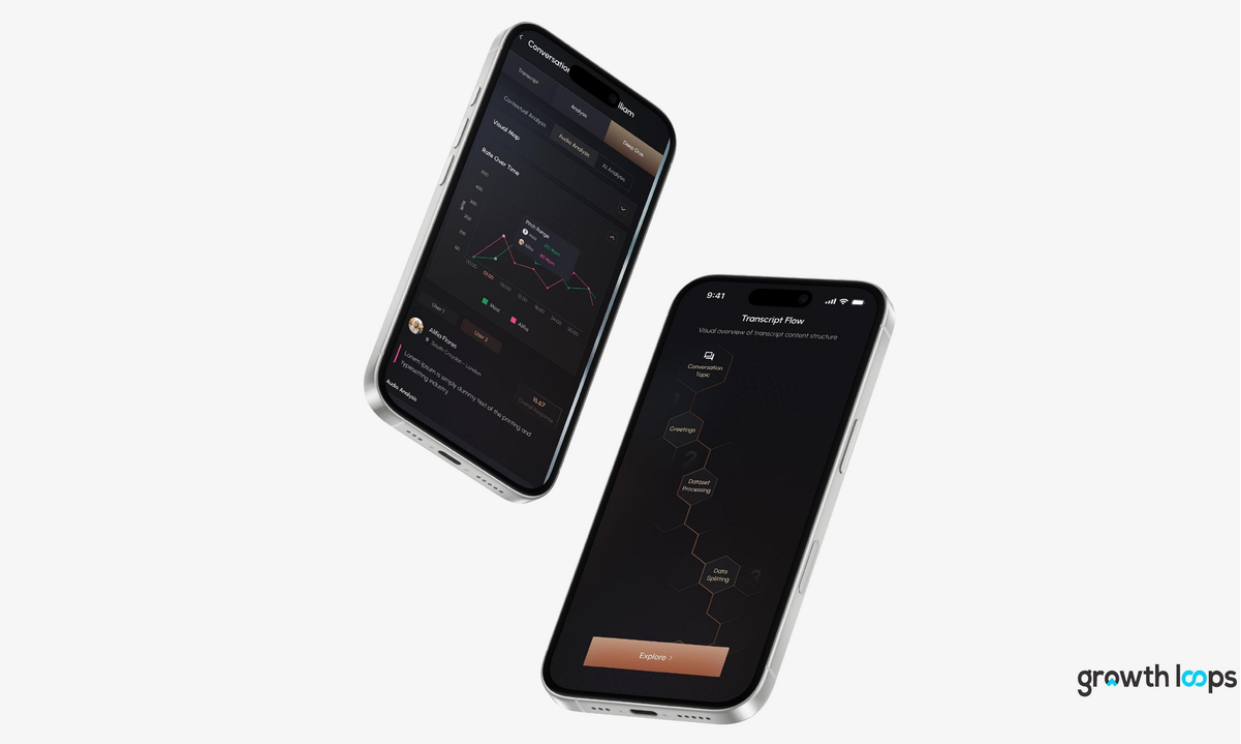
What was delivered

Growth Loops Technology delivered WingManAI from scratch to a production-ready platform encompassing two containerised backend services, a full PostgreSQL schema with 25+ models and 30+ migrations, a multi-stage async AI pipeline, and a three-tier monetised product. The AI analysis system covers 19 distinct conversation dimensions spanning language, acoustics, and relational dynamics.

The most technically demanding aspect was building the acoustic-linguistic pipeline: synchronising word-level audio features with transcribed speech at scale, then feeding those enriched representations into an LLM that could interpret them coherently. Getting structured, reliable outputs across diverse conversation styles, audio qualities, and

durations required significant iteration on prompt engineering, output schema design, and pre-processing robustness.

The consent architecture was equally challenging — demanding careful domain modelling. Professional conversations carry legal and personal sensitivity that consumer audio tools typically ignore. Building consent as a first-class entity in the data model shaped the entire downstream pipeline design.



Delivered screens — audio analysis with pitch/volume charts and the AI-generated transcript flow map

Key Engineering Takeaways

- Decouple compute-heavy AI workloads into separate services early — they scale and fail differently from API workloads
- Structured output from LLMs (Pydantic models) is essential for production — free-form responses are too fragile to write to a database
- Consent and privacy should be modelled in the schema from day one — retrofitting is painful and error-prone
- BullMQ with Redis is excellent for multi-stage pipelines, but job idempotency and status tracking must be built explicitly

- Acoustic features add genuine signal to conversation analysis, but only when correctly normalized across environments

Growth Loops Technology Pvt Ltd · Engineering Portfolio
gunendu@growth-loop.io · +91 9880806633