Transight Hackathon Problem Statement

Multimodal Omni-Channel Conversation Intelligence Backend

Background

Modern enterprises—especially in **banking, telecom, and customer support domains**—handle large volumes of customer interactions across **voice calls and digital text channels**, often in **multiple languages**. Deriving actionable insights from these conversations is critical for improving service quality, ensuring compliance, and managing operational risk.

With the availability of **multimodal AI models capable of directly understanding audio and text**, it is now possible to build intelligent backend systems that can analyze conversations without traditional speech-to-text or language-specific pipelines.

This hackathon challenges participants to design and implement a **backend-only, API-driven conversation intelligence system** that can process **voice or text conversations** and generate **structured, configurable insights** suitable for real-world enterprise integration.

Problem Statement

Design and implement a backend API system that can analyze multimodal customer conversations (voice or text) across languages and generate structured insights based on configurable client context and business rules.

The solution must be API-first and suitable for integration into mobile or web applications.

Scope & Functional Requirements

1. Input Handling (Multimodal)

- The system must accept **either**:
  - An audio file (customer support call recording), **or**
  - A text-based conversation transcript
- Language should be **auto-detected** by the system.

2. Core Conversation Intelligence (Mandatory)

Using a multimodal AI model, extract the following insights:

- Conversation summary
- Detected language(s)
- Overall sentiment or emotional tone
- Primary customer intent(s)
- Key topics or entities discussed

3. Configurable Client Context (Mandatory)

The system must support **client-defined configuration** that influences analysis.

Example (JSON-based configuration):

- Business domain (e.g., banking, telecom)
- Products or services
- Policies or rules
- Risk or compliance triggers

Participants may implement this as:

- JSON input
- Static configuration file
- Simple database schema

4. Advanced Analysis (Choose at Least One)

Implement **at least one** of the following:

- Compliance or policy violation detection
- Agent quality or performance scoring
- Call outcome classification (resolved, escalated, dropped, etc.)
- Risk or escalation score generation

5. Output Requirements

- Output must be a **well-structured JSON response**
- Designed as if it will be consumed by an enterprise application
- Must include both:

    - Analytical results
    - Any detected risks, flags, or classifications

---

6. Technical Expectations

- Backend-only solution (no UI required)
- Clear API endpoints
- Well-documented request/response formats
- Clean, readable, and maintainable code
- README explaining setup, design decisions, and assumptions

Optional Enhancements (Bonus)

- Multiple speakers handling
- Timeline-based sentiment or emotion tracking
- Extensible schema for adding new insight types
- Basic authentication or API key handling

Deliverables

1. Source code repository
2. API documentation (OpenAPI / Markdown)
3. Sample requests and responses
4. README explaining:
    - Architecture
    - AI usage approach
    - Configuration mechanism
    - Limitations and future improvements

Shortlisting & Evaluation Rubric

Category	Weight	What We Evaluate
Problem Understanding	20%	Clarity of interpretation, alignment with requirements, logical approach
API Design & Backend Architecture	20%	Endpoint design, request/response schemas, modularity, scalability thinking

AI Reasoning & Insight Quality	25%	Relevance, accuracy, and usefulness of extracted insights
Configurability & Flexibility	15%	How cleanly client context or rules influence analysis
Code Quality & Documentation	10%	Readability, structure, comments, and README clarity
Innovation & Practicality	10%	Real-world usefulness, thoughtful extensions, enterprise readiness

---

Intern Selection Criteria

Candidates demonstrating the following will be prioritized for the **AI Internship Program**:

- Strong problem decomposition and reasoning skills
- Ability to design production-ready backend APIs
- Practical application of multimodal AI models
- Clear thinking around enterprise use cases
- Clean code and documentation practices

Final Note to Participants

This challenge is not about building a perfect product in limited time—it is about **how you think, design, and apply AI to solve real enterprise problems**. Simplicity, clarity, and correctness are valued over complexity.

Wish you All the very best 👍