# Technical Design Document

**The proposed design is based on the modular, scalable, and fault-tolerant handle architecture** for an AI-powered voice system. There are some key phases of the architecture such as ingestion, processing, persistence, and observability. The architecture is able to provide high throughput under load, resilience in failure, secure storage for voice data and APIs support for the external usage.

## 1. Components and Roles

* **Ingress**

The user will connect via SIP/PSTN through a **Telephony provider** (e.g., Twilio). The provider generates a **webhook or audio URL** that will be forwards the stream into the system.

* **Edge**

**API Gateway/Load Balancer and WAF** are responsible for traffic distribution and protection. **Auth services** validates requests using API keys and signature. So, the entry point is “POST/voice-input” accepts audio URL or stream.

* **Processing Layer**

**The vice input service will v**alidate input, stores metadata, and enqueues jobs. The m**essage queue** buffers transcription requests and ensures reliable delivery. DLQ supports the failed jobs handling and **STT workers** autoscaling group that pulls jobs, processes audio to text, and stores transcripts. In the **sentiments workers** autoscale group that consumes transcripts and performs sentiment analysis.

* **Persistence**

Metadata database stores transcripts, job status, and sentiment results. In encrypted blob storage system stores raw audio files. The KMS/Key Store only manages encryption keys.

* **Observability & Ops**

The m**onitoring/metrics** are responsible to tracks latency, errors, and throughput. **Dead Letter Queue (DLQ)** stores failed jobs for operator review. Triggers notifications for failures or anomalies are handled by alerts/pager.

* **Clients**

Via **callback/polling dashboard** clients are notified via webhook or can query job status via APIs.

## 2. API Endpoints

1. POST/voice-input
   * It accepts audio stream/URL.
   * Then validates request, stores metadata, and enqueues transcription job.
   * Finally returns job\_id and processing status.
2. POST/transcribe
   * In the internal endpoint used by STT workers.
   * It’s functionality to converts audio to text and updates transcript in database.
   * Finally returns transcript.
3. POST/analyze-sentiment
   * It consumes transcript, runs sentiment analysis.
   * Finally returns sentiment classification such that positive, neutral, and negative etc.

## 3. Scalability and Fault Tolerance

* **Scalability**:
  + - API Gateway and Load Balancer distributes incoming traffic and can enhance.
    - STT & Sentiment are autoscaled to handle varying load.
    - Message Queue decouples consumers for elastic scaling.
* **Fault Tolerance**:
  + - Handle the redundant components (API Gateway, DB replicas, Blob storage).
    - DLQ guarantee failed jobs are isolated and then retried manually.
    - Persistent queues prevent data loss if workers failed.
    - Client callbacks/webhooks support retries on network errors or failure.

## 4. Security Strategy

* **Data Encryption**:

**All the a**udio files are stored in **encrypted blob storage**. The transcripts and metadata encrypted at rest in **metadata database**. The encryption keys managed by **KMS/Key Store**.

* **Authentication & Authorization**:

**The** API requests are secured with API keys and signatures. And the HTTPS enforced for all external communication.

* **Data Protection**:

The system able to secure handle sensitive audio and transcript data. Access controlled by role-based policies and roles.

-------------------------------------------