II. Technical Design Document

1) Queue configuration overview

Service 1 : Call Transcription

Queue : transcription_queue

- Ce service s'occupe de la transcription de l'audio des appels en texte. Il reçoit des messages contenant des métadonnées d'appel et le fichier audio à transcrire.
- **Comment ça marche**:
  - Le service de réception des appels ou un autre service envoie les informations de chaque appel à cette queue. Ces informations incluent l'ID de l'appel, le fichier audio et d'autres détails nécessaires à la transcription.
  - Le service de transcription prend ces messages, les lit et commence à transcrire l'audio en texte.

Service 2 : Call Feature Extraction

Queue : feature_extraction_queue

- Après que le texte a été transcrit par le premier service, le service suivant extrait des informations utiles, comme les mots-clés ou les événements importants du texte de l'appel.
- **Comment ça marche**:
  - Le service de transcription, une fois la transcription terminée, pousse le texte dans cette queue.
  - Le service d'extraction des caractéristiques consomme alors les messages de cette queue et effectue des analyses pour en tirer des éléments intéressants.

Service 3 : Sentiment Analysis (Exige la fin du Service 2)

Queue : sentiment_analysis_queue

- Ce service analyse les émotions ou l'humeur présentes dans le texte extrait des appels pour savoir si l'appelant est en colère, satisfait, frustré, etc.
- **Comment ça marche**:
  - Une fois l'extraction des caractéristiques terminée, les données pertinentes sont envoyées à cette queue.
  - Le service d'analyse des sentiments prend ces messages, les analyse et détermine l'état émotionnel de l'appelant.

Service 4 : Customer Intent Classification

Queue : intent_classification_queue

- **But** : Après avoir déterminé l'émotion de l'appelant, ce service classifie l'intention de l'appel : est-ce un appel de support, une demande de renseignements, une réclamation, ou autre chose ?

- **Comment ça marche**:
  - Après l'analyse des sentiments, le service pousse les résultats de l'analyse, ainsi que d'autres informations pertinentes (comme le texte analysé) dans cette queue.
  - Le service de classification des intentions consomme ces messages et détermine l'intention de l'appel.

Utilisation du Pub/Sub pour les Services 3 et 4 :

Les services **Sentiment Analysis** et **Intent Classification** utilisent tous les deux le modèle Pub/Sub via **Redis** pour leur communication.

2) Error handling and retry approach:

Service 5 : Error Management Service (EMS)

Objectif du Service :

L'**Error Management Service (EMS)** est conçu pour centraliser et gérer toutes les erreurs générées par les autres services de l'architecture. Il reçoit les erreurs de chaque service, prend des décisions sur la manière de traiter chaque message erroné, et s'assure que les erreurs sont soit corrigées (réessayées), soit redirigées vers une queue de type "dead-letter" pour une investigation manuelle ultérieure.

Responsabilités de l'EMS :

Objectif du Service :

L'**Error Management Service (EMS)** est conçu pour centraliser et gérer toutes les erreurs générées par les autres services de l'architecture. Il reçoit les erreurs de chaque service, prend des décisions sur la manière de traiter chaque message erroné, et s'assure que les erreurs sont soit corrigées (réessayées), soit redirigées vers une queue de type "dead-letter" pour une investigation manuelle ultérieure.

- **Centraliser les erreurs** : EMS intercepte toutes les erreurs provenant des autres services et les centralise dans une queue dédiée.
- **Analyser et classer les erreurs** : L'EMS analyse les erreurs et décide si l'erreur est réparable, si elle nécessite un réessai, ou si elle doit être placée dans une dead-letter queue.
- **Réessayage automatique** : Pour les erreurs temporaires, EMS réessaie de traiter le message après un délai exponentiel.
- **Gestion des erreurs critiques** : Si une erreur ne peut pas être corrigée après plusieurs tentatives, EMS envoie le message dans une **dead-letter queue** pour un traitement manuel.

- **Réacheminement des messages** : Si le message peut être corrigé ou réessayé, EMS le redirige vers la bonne queue pour un traitement ultérieur.

3)Message format overview

Champs du Message :

- **message_id** : Identifiant unique pour chaque message. Cela permet de suivre chaque message tout au long du processus.
- **timestamp** : L'heure à laquelle le message a été généré ou traité. Cela permet de suivre l'historique des messages.
- **source_service** : Nom du service qui a produit le message (par exemple : "Call Transcription").
- **payload** : Contenu principal du message. C'est ici que les données spécifiques du service sont stockées (par exemple, les transcriptions des appels, les résultats d'analyse des sentiments, etc.).
- **error** : Champ qui contient une description de l'erreur, le cas échéant. Si le traitement a échoué, ce champ indiquera l'erreur associée.
- **status** : Le statut actuel du message. Par exemple, pending, in_progress, processed, failed, etc.
- **retry_count** : Nombre de tentatives effectuées pour traiter ce message. Cela est utilisé pour déterminer si un message doit être réessayé ou déplacé vers une dead-letter queue.

Exemples:

Call Transcription → Call Feature Extraction:

```
{
  "message_id": "call_12345_transcription",
  "timestamp": "2024-11-21T10:05:00Z",
  "source_service": "Call Transcription",
  "payload": {
    "call_id": "call_12345",
    "audio_url": "https://example.com/audio/call_12345.wav",
    "transcription": "Hello, this is a call from customer support..."
  },
  "error": null,
  "status": "pending",
  "retry_count": 0
}
```


Exemples de Message avec Erreur :


```
{
  "message_id": "call_12345_sentiment_analysis",
  "timestamp": "2024-11-21T10:15:00Z",
  "source_service": "Sentiment Analysis",
  "payload": {
    "call_id": "call_12345",
    "transcription": "Hello, this is a call from customer support..."
  },
  "error": {
    "message": "Failed to process sentiment analysis due to missing data",
    "code": 500,
    "details": "Sentiment analysis data was incomplete or corrupt"
  },
  "status": "failed",
  "retry_count": 2
}
```

4) Service communication definition:

Les services de notre système utilisent principalement la communication suivante :

- Producteur-Consommateur (Producer-Consumer) : Un service produit des messages qui sont ensuite consommés par un autre service. Ce modèle permet de gérer efficacement les tâches longues ou gourmande en ressources, telles que la transcription d'appel, l'analyse de sentiment, ou la classification des intentions.
- Publish-Subscribe (Pub/Sub) : Ce modèle est utilisé pour les services parallèles, où plusieurs services abonnés à un même sujet peuvent réagir à des messages spécifiques. Par exemple, les services d'analyse des sentiments et de classification des intentions sont abonnés à un même canal pour traiter les informations dans un flux parallèle.
- Le protocole WebSocket est utilisé pour les communications bidirectionnelles et en temps réel entre certains services.