Real-Time AI-Powered Voice Translator

Context

As part of the Google Cloud Hackathon, the objective is to design a real-time voice translator using advanced artificial intelligence technologies.

This project will rely on Google Cloud and its tools (such as Vertex AI and Cloud AI APIs) to implement an innovative and functional system that accurately translates voice into multiple languages.

Project Objectives

### Main Objectives:

* Develop an application capable of translating voice in real-time into multiple languages.
* Leverage existing AI models on Vertex AI to minimize development time.
* Implement the following technologies:
  + Speech-to-Text (voice recognition).
  + Text-to-Text (text translation).
  + Text-to-Speech (voice synthesis).

### Secondary Objectives:

* Integrate a communication server to transmit translations to external listening devices.
* Provide educational feedback for users wishing to improve their diction or learn a new language.

### Innovations :

* Integration with generative models like Gemini to enrich translations by adding context.
* Deployment of an intuitive and attractive user interface (UI).

## Attendus Fonctionnels

Real-time recognition and translation with minimal latency.

Linguistic flexibility: allow users to choose input and output languages.

User Interface (UI): simple, providing real-time visual feedback of translations

### Educational features:

* Suggestions to improve diction.
* Grammar indications in the translated language

Interoperability: the system must be compatible with various listening devices.

Technical Requirements

Software Components

### Google Cloud Services :

* + Speech-to-Text API.
  + Translate API.
  + Text-to-Speech API.

### Langages :

* + Python ((integration with Google APIs and UI).

### Bibliothèques :

* + sounddevice for audio stream management.
  + queue for synchronizing processing steps.

### Performance Requirements:

* + Total processing time (voice → text → translation → voice) under 2 seconds.
  + Stable audio stream management, even with interruptions or silences.

### Architecture :

* + A system based on a parallel pipeline to simultaneously manage different steps (capture → transcribe → translate → synthesize).
  + Automatic restart capability upon detecting silence.

### Modifiable Parameters:

* + Input and output languages.
  + Silence detection timeout.

### Scalability :

* + Optimized to work on constrained environments (standard laptops or cloud servers).
  + Extensible features to include other languages or dialects.

Non-Functional Requirements

### Sécurity :

* Audio streams must not be recorded or shared without explicit consent.
* Data must be processed in a secure Google Cloud environment.

### Accessibility :

* User interface accessible to individuals with visual or hearing impairments.

### Robustness :

* Resistant to network errors or audio input interruptions.