# **Reference Implementation for a Speech Translation App Using Google’s Speech Translation Services**

The POC for the application being demonstrated uses a microservice that receives an audio message, translates the message to a set of predefined languages, and stores the translated messages in audio files. The client Android app downloads and plays the translated audio files at the user request.

The Solution includes the following Components

## Microservice

The microservice is implemented on Cloud Functions for Firebase and uses the following Cloud AI products to translate the messages:

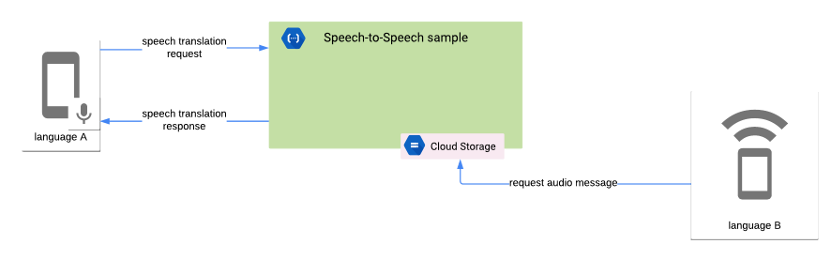
* Cloud Speech-to-Text
* Cloud Translation
* Cloud Text-to-Speech

The microservice stores translated audio messages in a bucket in Cloud Storage for Firebase.

## Client App

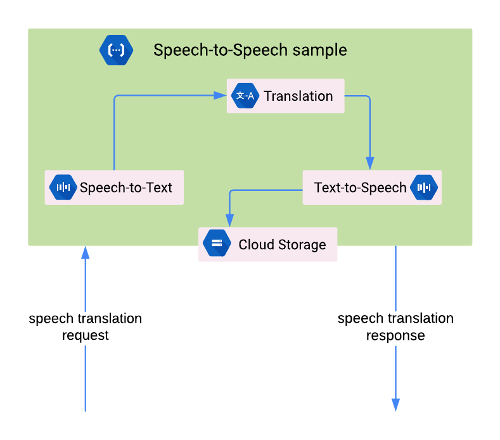
The client component is an Android app that records audio messages and downloads the translated messages from the Cloud Storage bucket.

The following diagram shows the interaction between the microservice and the client app:



The microservice performs the following tasks:

1. Receives the audio message in the Base64 encoded format
2. Transcribes the audio message using the Cloud Speech-to-Text API.
3. Translates the transcribed message using the Translation API.
4. Synthesizes the translated message using the Text-to-Speech API.
5. Stores the translated audio message in a Cloud Storage bucket.
6. Sends the response back to the client. The response includes the locale of the translated audio message.



The POC utilizes Cloud Functions for Firebase to build a microservice that encapsulates the logic required to translate audio messages using the following Cloud AI products:

* Cloud Speech-to-Text API
* Translation API
* Cloud Text-to-Speech API

It uses the Android Framework APIs to record audio to provide audio data to the Cloud Speech-to-Text API. It further uses the Cronet Library to upload speech data from the client app to the microservice and to download translated messages from Cloud Storage.

A brief overview of the factors that contribute to the cost of developing a full fledged Speech to Speech to translation

* Firebase defines quotas for Cloud Functions usage that specify resource, time, and rate limits. For more information, [Quotas and Limits](https://firebase.google.com/docs/functions/quotas) in the Firebase documentation.
* Cloud Speech-to-Text API usage is priced monthly based on the length of audio successfully processed. For more information, see [Cloud Speech-to-Text API Pricing](https://cloud.google.com/speech-to-text/pricing).
* Translation API usage is priced monthly based on the amount of characters sent to the API for processing. For more information, see[Translation API Pricing](https://cloud.google.com/translate/pricing).
* Text-to-Speech API usage is priced monthly based on the amount of characters to synthesize into audio. For more information, see[Text-to-Speech API Pricing](https://cloud.google.com/text-to-speech/pricing).
* Firebase Storage usage fees are processed as Google Cloud Storage fees. For more information, see [Cloud Storage Pricing](https://cloud.google.com/storage/pricing).
* Text to Speech services for Arabic are currently not available for Google. For a list of languages supported by Google see [Link](https://cloud.google.com/text-to-speech/docs/voices) Incidentally Google had anounced in 2014 that it would never support Arabic (see [link](https://en.wikipedia.org/wiki/Google_Text-to-Speech)). Other solution providers like Microsoft Speech Services, Amazon Polly, IBM Watson do not support it either. Speech synthesis for Arabic requires an end-toend neural network with a large corpus of text to train the model. It requires the implementation of [TensorFlow](https://www.tensorflow.org/) (An open source machine learning framework), [Librosa](https://librosa.github.io/librosa/)(A python package for music nd audio analysis), tqdm and maptolib (statistical tools to measure the progress of TTS quality of learning). In general most Speech service providers do not provide TTS services for no more than a dozen langauges. However Google seems to be leading the pack with over 80 langauges.
* To augument the features of the application, it requires a high performance, high-availability, asynchronous real-time messaging platform that allows users (endpoints) to communicate with each other in real-time. While this can be achieved using Google Firebase. It does not scale well when the platform reaches a threshhold of more than a few 1000 users. In such cases the app will have to piggy back on athird party messaging platform. Depending on the platform being chosen, their prices for services vary. Examples are Sendbird, Twilio, Pubnub, Mesibo etc.

## A Note on Firebase

In this demo we have made use of Cloud Firestore as the datastore. Since this is a demo, we have a used a very basic data structure.

Cloud Firestore is a NoSQL JSON data store. Essentially, everything in the Cloud Firestore is a JSON object, and each key of this JSON object has its own URL. A sample of our data is represented in the image below

Cloud Firestore [favors a denormalized data structure](https://firebase.google.com/docs/firestore/manage-data/structure-data), so it’s okay to include senderId and senderName for each message item. A denormalized data structure means you’ll duplicate a lot of data, but the upside is faster data retrieval.

"channels": [{

"MOuL1sdbrnh0x1zGuXn7": { // channel id

"name": "Verbat",

"thread": [{

"3a6Fo5rrUcBqhUJcLsP0": { // message id

"content": "Can you hear me",

“Lang”: “En”,

"created": "Feb 20, 2019 at 10:44:11 PM”,

"senderID": "YCrPJF3shzWSHagmr0Zl2WZFBgT2",

"senderName": "Shibu",

},

"4LXlVnWnoqyZEuKiiubh": { // message id

"content": "Hello there",

“Lang”: “En”

"created": "Feb 20, 2019 at 10:40:05",

"senderID": "f84PFeGl2yaqUDaSiTVeqe9gHfD3",

"senderName": "Rikhil",

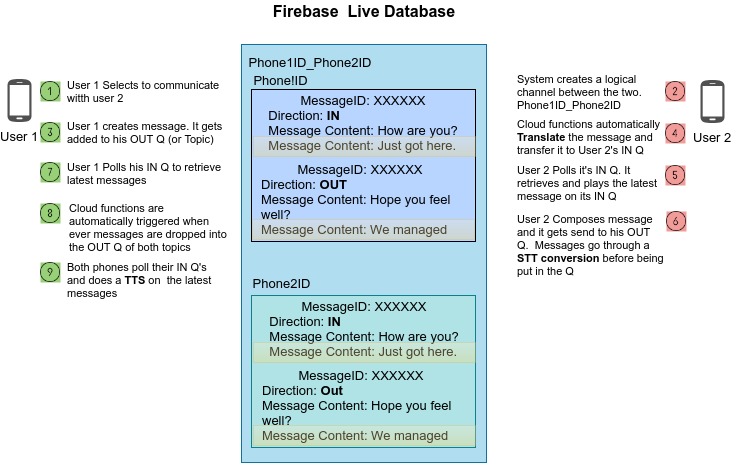
},

}]

},

}]

}



In situations where a direct TTS conversion is not available, the speech can be translated to the native script of the target user. For example, in a communication between a French resident and a Saudi resident, the French resident speaks in French. The text gets transcribed, translated and gets relayed to the Saudi resident in native script (Locale based Arabic phrase) to a high degree of accuracy. The Saudi resident responds to the received message via speech. The process repeats, with the exception that, since the text to speech service is available for French, it gets relayed as an audio message. Effectively, even though the TTS service is not available for Arabic, it does not stop an Arabic resident from communicating effectively with his counterparts around the world.

## Performance Considerations

The reference implementation lags by a few seconds for voice chat services. This is becasue, we’ve used a web service implementation. This causes a round trip delay between Google speech services and the target devices since all messages have to be brokered by the web service. In a production environment, the devices talk to google cloud functions, which is an event driven serverless computing platform. Cloud functions can access all google services (firebase, online cloudstore, messaging etc) like it were a local resource. This could be an effective strategy for reducing lags.