# Designing a scalable Conversational Interfaces

**Cristian Gutierrez**[1§], **Darshi Kasondra**[1§], **Archana Ravi**[1§], **Mounica Dingari**[1§]

[1]California State University of Northridge

*Additional work related to this paper can be fournd at https://github.com/thehamop1/AdvSoftwareEngineeringProject;*

*This version dated May 9, 2021*

**Abstract** The way we use interact with systems is contantly evolving with new advances in technology. As buisness continue to need flexible user interfaces many of them are implementing conversatinal interfaces in order to fufill customer needs without having to require users to learn how to use an entirely seperate application in order to make transactions. Modern conversatinal interfaces are fairly powerful with recent adavancements in natural language processing, integration with mobile asistants, and flexible frameworks. In this paper we provide background to some of the concepts of conversatinal interfaces, examing the current state of commercially available solutions, and provide a reference architecture for implementing a system centered around a conversatinal interface.

***For correspondence:**
cristian.gutierrez.56@my.csun.edu (CG)
**Present address:** [§]Department of Engineering and Computer Science, California State University of Northridge, United States of America

## 1 Introduction

### 1.1 Previous Work, Methods, Procedures

There is a growing need for approchable user interfaces as more interactions become digital. For example banking, transactions, and flights are all largely growing to be digital. In order to easily accessed and usable for all types of users conversatinal interfaces are the most ideal due to their low learning curve. If a user can use their native language in order to complete actions on any given system the need for 24/7 support or complex user interfaces become obsolete. This allows buisness to keep user satisfaction high while keeping costs low. Additionally it provides several benefits to users as they are able to access data from large databases easily, complete transactions even if their not so tech savy, have multilingual support, and have 24/7 support.

### 1.2 Previous Work, Methods, Procedures

The development and interest of conversatinal interfaces was has been a subject of interest since the 1970's. Early examples of these early chatbots include ELIZA, ALICE, and PARRY. Many of these early chatbots worked with the use of simple pattern matching. This simple regular expression based matching was combined with a tree design for controlling the flow of conversations. One of the major drawbacks of this naive form of design was the frequent matching of user utterances with conversation points that happened further up the tree. This would often lead to looping conversations. Researchers at the time had to develop markup based langugues such as AIML in order to develop expert systems. These large complex forms of nested databases had to be constantly maintined in order to add new features to these chatbots. With the modern development of machine learning algorithms in order to parse and extract meaning from user utterances. Many com-

mercial solutions for developing conversatinal interfaces are now widely available and come with many integrations for various platforms.

### 1.3 Background

There are some basic concepts that are universal to most conversatinal interface platforms. The first is the user utterance which can either come in the form of a text entry or speech with the use of a microphone. Additional signal processing is required to transcribe the audio signals into text. This text is usually normalized where all text has its puntuation removed and all letters are moved into the same letter case. Next depending on the platform various machine learning algorithms are used in order to extract certain key tokens from the string. The most intent which is the objective of the user. For example the utterance "What is the weather in Los Angeles" would have the intenet of weather. This is a topic that the application would have to be designed to respond to. Next would be entities which are certain tokens the application will use to complete requests once an intenet has been deduced.

## 2 Implementation Guide

### 2.1 Use Cases

The following implementation guide was designed with the assumption that it would be used in a buisness context such as e-commerce site, resturant, or customer support. Additionally little sensitive information would be transmitted between the conversatinal interface and the service could be an addition to existing buisness infrastructure. Additionally this Implementation guide is more suitable for small to medium sized buisness who are looking to provide the user with the ability to retrieve basic data and transactions.

An example use case for this reference architecture would be a small chain of resturants. In this scenario a customer may want to make reservations for instance while the resturant is closed. In this scenario integrating a conversatinal interface on the buisnesses website and mobile assistant would provide the user with the ability to create or cancel reservations while the buisness is currently closed. Even customers with little knowledge of how to browse the internet or create reservations online would be able to interact with their mobile asistants in order to interact with the system.

### 2.2 Technical Stack

For this particular implementation guide a stack such as ReactJs, SQL, nodejs, dialogflow, and Google Assistant/Siri would be used. NodeJS would be used on the lambda functions and buisness enterprise. Any in buisness databases would be created with an SQL flavor and an ORM such as knex.js would be used to store various information. If a buisness did not have an existing website it would be created in ReactJS, alternatively if an existing website was already available a simple iframe based solution could be used from the existing dialogflow API integrations.

As stated before any interaction needed with existing buisness infrastructure could be handled with lambda funcitons. The existing library most supported by the Dialogflow API is NodeJS. It is extermly cost effective and easy to integrate using Google Cloud Functions. In order to full take advantage of deploying to various platforms different buisness logic needs to created to interact with various front-ends. The Dialogflow API allows you to create specific reponses based on the clients platform in order not to overwhelm the client with long responses or provide more relevant information to the user.

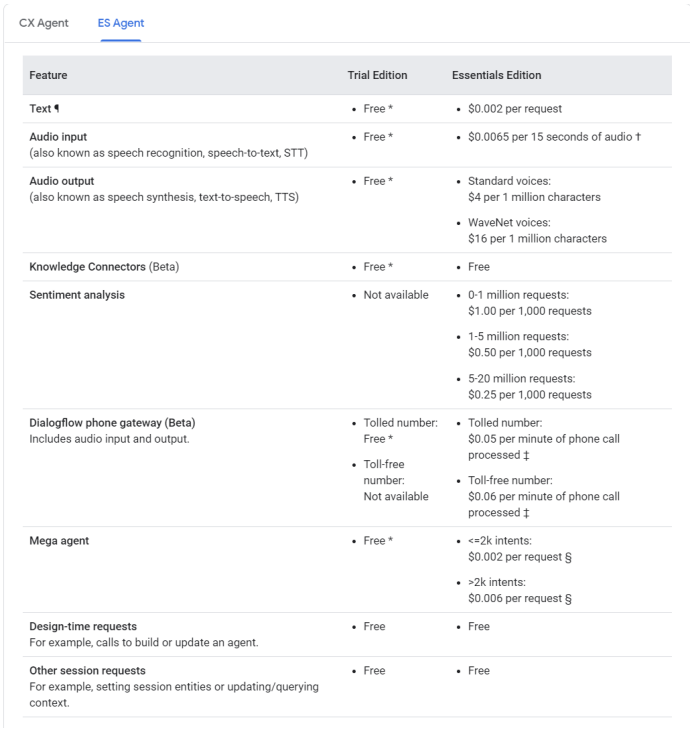
An added benefit to using lambda functions is that all of the heavy machine learning processing is carried out on the Dialogflow API side so no heavy processing is required within the runtime of the lambda itself. This allows the call from the client to be quickly parsed via a RESTful API call and intenets, entities, and context can already be deduced ahead of time. By allowing the buisness logic to be compeltley free of all machine learning it allows the particular buisness implemnting the system save money on renting a fully dedicated server to run the machine learning algorithm. Keep in mind that although the use of lambda functions reduces cost there is an additonal cost required for every call to the Dialogflow service. Additionally the version of the DialogFlow API used also factors into cost. If the complexity of the conversations are relativley simple version 2 of the API can be used to increase savings on this implementation.

Lastly in terms of getting a voice enabled conversatinal interface for this systmem would require the use of the newer DialogFlow V3 API or the use of a deprecated API in order to interface wtih the older DialogFlow version 2. Although this option is available using the newer nodejs package it is highly discouraged as this part of the library is no longer receiveing updates. Additionally in order to connect this service to iOS an additionial layer of buisness logic has to be placed between the dialogflow API calls and the Siri front-end. The buisness should consider the development cost required in order to keep iOS integration supported versus how benefical it is from a buisness stand-point to keep it supported.

### 2.3 Cost Benefit Analysis

In order to understand the on-going cost of having lambda function costs and DialogFlow API calls the following charts should be considered when considering the benefit of adding

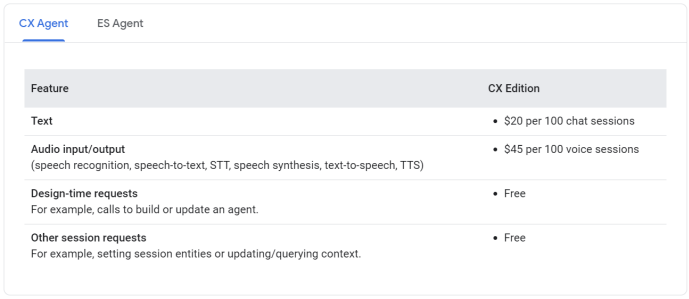
**Figure 1.** DialogFlow Version 2



CX Agent ES Agent

| Feature | Trial Edition | Essentials Edition |
|---|---|---|
| **Text** ¶ | • Free * | • $0.002 per request |
| **Audio input** (also known as speech recognition, speech-to-text, STT) | • Free * | • $0.0065 per 15 seconds of audio † |
| **Audio output** (also known as speech synthesis, text-to-speech, TTS) | • Free * | • Standard voices: $4 per 1 million characters<br>• WaveNet voices: $16 per 1 million characters |
| **Knowledge Connectors** (Beta) | • Free * | • Free |
| **Sentiment analysis** | • Not available | • 0-1 million requests: $1.00 per 1,000 requests<br>• 1-5 million requests: $0.50 per 1,000 requests<br>• 5-20 million requests: $0.25 per 1,000 requests |
| **Dialogflow phone gateway (Beta)** Includes audio input and output. | • Tolled number: Free *<br>• Toll-free number: Not available | • Tolled number: $0.05 per minute of phone call processed ‡<br>• Toll-free number: $0.06 per minute of phone call processed ‡ |
| **Mega agent** | • Free * | • <=2k intents: $0.002 per request §<br>• >2k intents: $0.006 per request § |
| **Design-time requests** For example, calls to build or update an agent. | • Free | • Free |
| **Other session requests** For example, setting session entities or updating/querying context. | • Free | • Free |

**Figure 2.** DialogFlow Version 3



CX Agent ES Agent

| Feature | CX Edition |
|---|---|
| **Text** | • $20 per 100 chat sessions |
| **Audio input/output** (speech recognition, speech-to-text, STT, speech synthesis, text-to-speech, TTS) | • $45 per 100 voice sessions |
| **Design-time requests** For example, calls to build or update an agent. | • Free |
| **Other session requests** For example, setting session entities or updating/querying context. | • Free |

a converstaional interface to an e-commerce site.

# References