Conversational Bot for Banks

Authors: Shreya Anand   
Visibility: Across Alphabet   
Originally Proposed: 2021-06-15 / Last Updated: 2021-08-18

Objective

A conversational bot for the banking sector which aims at tackling the customer acquisition and retention use-case by providing 24X7 support to customers enabling them to perform common banking tasks anytime and anywhere. Built using Dialogflow and integrated with Google assistant, the aim is to provide customers with seamless banking experience and at the same time, save the banking staff from conversational overhead with the customers.

# Requirements

* Understanding of the basic terminologies associated with a DialogFlow Agent

<https://cloud.google.com/dialogflow/es/docs/basics>

* Using Node.js with GCP

<https://cloud.google.com/nodejs/docs/>

* Using Firestore

<https://firebase.google.com/docs/firestore>

* Using Cloud functions

<https://cloud.google.com/functions>

* Integrating Assistant with Dialogflow

<https://cloud.google.com/dialogflow/es/docs/integrations/aog#:~:text=Dialogflow%20provides%20a%20seamless%20integration,privacy%2C%20support%2C%20and%20SLAs>

# Inspiration for the Project

Since the pandemic, many banks have experienced a [200% rise](https://www.cnbc.com/2020/05/27/coronavirus-crisis-mobile-banking-surge-is-a-shift-likely-to-stick.html) in mobile app registrations which is expected to grow more in the near future. Now, with this, there has been a focus on making the digital experience smoother for the customers, especially with AI. Consider the less accessible geographical areas, where there are no physical branches present, for the people there, being able to use voice commands to, for instance, pay their bills can be rewarding. And even for people like us who do visit the branch, there will be less friction if we get our queries answered through automation and not have to wait in long queues inside the bank.

Taking the example of a banking customer in India, who is coming from a less educated background, doesn’t understand English, or Hindi, and is comfortable only in one of his regional languages. What can be better than having AI powered conversational systems to let him perform the common banking activities in his native language, over voice activated commands. These can be installed at ATMs or even physical branches of banks so that the repetitive mundane tasks can be performed through AI automation without the need to wait for another human agent to respond to us. These also save the banking staff from stressful situations that arise from direct communication.

# Overview

Dialogflow is the solution offered by Google which uses Machine Learning to give customers a rich conversational experience. It supports a diverse set of use cases from Customer service to home entertainment.

For this project, I have built a chatbot for banks using Dialogflow Essentials. Added 20+ intents which correspond to the common tasks of the banking sector along with relevant query resolution intents as well. I defined a node.js service and deployed it on cloud functions - this basically is to enable the agent to interact with a database. For the data, I created a sample collection inside the firestore db, to replicate a customer dataset for a bank - containing information like their account number, name, bank balance etc. Lastly, I integrated it with Google assistant.

# Architectural Diagram

# Implementation Details

## Created a project -

To create a project, you must have the **resourcemanager.projects.create** permission. For information on how to grant individuals the role and limit organization-wide access, see the [Managing Default Organization Roles](https://cloud.google.com/resource-manager/docs/default-access-control) page. Note : Each project can have only one agent.

To create a new project,

1. Go to the **Manage resources** page in the Cloud Console.  
   [Go to the Manage Resources page](https://console.cloud.google.com/cloud-resource-manager)
2. On the **Select organization** drop-down list at the top of the page, select the organization in which you want to create a project. If you are a free trial user, skip this step, as this list does not appear.
3. Click **Create Project**.
4. In the **New Project** window that appears, enter a project name and select a billing account as applicable. A project name can contain only letters, numbers, single quotes, hyphens, spaces, or exclamation points, and must be between 4 and 30 characters.
5. Enter the parent organization or folder in the **Location** box. That resource will be the hierarchical parent of the new project.
6. When you're finished entering new project details, click **Create**

<https://cloud.google.com/resource-manager/docs/creating-managing-projects#console>

## Enabled the API -

You must enable the *Dialogflow API* for your project. For more information on enabling APIs, see the [Service Usage documentation](https://cloud.google.com/service-usage/docs/enable-disable).

## Setup Authentication -

* Create a Service account
  + A service account provides credentials for applications, as opposed to end-users. When an identity calls an API, Google Cloud requires that the identity has the appropriate permissions. You can grant permissions by granting *roles* to a service account.
  + <https://cloud.google.com/iam/docs/creating-managing-service-accounts>
* Create the service account Key
  + <https://cloud.google.com/iam/docs/creating-managing-service-account-keys>
  + This key will be used in the fulfillment section, to connect the dialogflow to the database backend that I have created using firestore.
* Granting necessary roles to Service account
  + Grant the following roles to your service account :
    - Dialogflow API Admin
    - Dialogflow API Client
    - Dialogflow Service Agent
    - Cloud Functions Invoker
  + <https://cloud.google.com/iam/docs/granting-changing-revoking-access>

## Created the Dialogflow Agent - FinanceDemo

* + Go to the [Dialogflow ES Console](https://dialogflow.cloud.google.com/).
  + If requested, sign in to the Dialogflow console. See [Dialogflow console overview](https://cloud.google.com/dialogflow/docs/console) for more information.
  + Click **Create Agent** in the left menu.
  + Enter your agent's name, default language, default time zone and GCP project, then click the **Create** button.
  + If you wish to make a multilingual bot, use the left sidebar menu, click the add language button next to the existing language(s). This takes you to the **Languages** tab of agent settings. Choose a language(currently supports only Hindi) from the **Select Additional Language** dropdown menu.

## Created a dummy banking dataset -

* Created a firestore collection - <https://cloud.google.com/firestore/docs/quickstart-servers>
* Added dummy data to it using the console, the dataset contained the following values - Banki Id, Customer Name, Current Balance, electricity bill, wifi bill, last transaction date, atm pin number., etc.

## Wrote the fulfillment code using node.js -

* Set up the template to use dialogflow intents using node.js <https://cloud.google.com/firestore/docs/quickstart-servers#initialize>
* I have created various functions, each to handle a particular intent of dialogflow, which include adding data to the database, accessing data, and updating the data that is already there.

<https://cloud.google.com/firestore/docs/quickstart-servers#add_data>

<https://cloud.google.com/firestore/docs/quickstart-servers#read_data>

<https://cloud.google.com/firestore/docs/quickstart-servers#next_steps>

* Then, after creating the functions, I called the function when the agent matching its action was called. To learn more about how to work with fulfillments in dialogflow - visit <https://cloud.google.com/dialogflow/es/docs/fulfillment-overview>
* The functions were then uploaded to cloud functions - I used the webhook service to call the functions every time a user interacts with the agent with a query. An inline editor can also be used instead, which automatically uploads the function code to cloud functions under the mentioned project. <https://cloud.google.com/dialogflow/es/docs/fulfillment-inline-editor>

<https://cloud.google.com/dialogflow/es/docs/fulfillment-webhook>

* Also, if using webhook, the mode should be enabled for each intent and subintents inside the dialogflow console.

## Uploading code on Cloud Functions -

Then, all the code has to be uploaded to cloud functions with the url mentioned under the webhook call category on the dialogflow console to invoke the webhook service.

<https://cloud.google.com/functions/docs/first-nodejs>

## Integration with Google Assistant -

The final step in my project was to integrate the bot with google assistant. Then after testing the app, I made the alpha deployment so that the bot could be tested on devices which have a google assistant installed on it.

<https://cloud.google.com/dialogflow/es/docs/integrations/aog#:~:text=Dialogflow%20provides%20a%20seamless%20integration,privacy%2C%20support%2C%20and%20SLAs>

# Link to Code

The following is the link to the code which contains the entire code base used in the project.

<https://github.com/shreyaanand29/Dialogflow-Banking-Chatbot>

**Note:** The repository does not contain the credentials.json file which stores the keys for authentication of invocation made from the service account invoking the function code uploaded on cloud functions. This is in adherence to the security norms and the user is expected to use his/her own credentials for the same.

# Next Steps

Currently, the Bot can handle conversations related to common banking transactions - like creating an account, checking the current bank balance, adding money, withdrawing money., etc and also some common queries that the customers might have related to the banking sector - like how to apply for a loan, what documents are required for the process of KYC., etc.

The next steps from here could be coming up with a robust security architecture so that this bot can be integrated with the present banking system - at ATMs or the physical branches.

Also, the width can be expanded to cover more banking related tasks and more query resolution intents.