Dr B R Ambedkar National Institute of Technology, Jalandhar

G T Road Bye Pass, Jalandhar – 144011 (Punjab)

Department of Electronics and Communication

# PROJECT REPORT

SUMMER INTERNSHIP-CUM-TRAINING

JUNE-JULY (2019)

Mohammad Anas 16104045

Electronics and Communication Engineering

# TO DEVELOP A CHATBOT & integrate it with the following API's of the Meridian Project:

- **Simple Estimate**
- **Retail Estimate**
- **Medical Services**
- **Medical Service Components**

**Name:** Mohammad Anas

**Roll No:** 16104045

**Discipline:** Electronics and Communication Engineering

**Name of the Organization:** UnitedHealth Group (UHG) - Optum Global Solutions,

Hyderabad, Telangana, India.

**Name of the Institute:** Dr B R Ambedkar National Institute of Technology, Jalandhar, Punjab, India.

# CONTENTS

# ACKNOWLEDGMENT

***The internship opportunity I had with UnitedHealth Group (UHG) – Optum Global Solutions was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me though this internship period.***

***Bearing in mind previous I am using this opportunity to express my deepest gratitude and special thanks to Mr. M.S.Srinath, Product Owner, who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my project at their esteemed organization and extending during the training.***

***I express my deepest thanks to Mrs. Srujana Sreevastava, Product Manager, for taking part in useful decision & giving necessary advices and guidance and arranged all facilities to make life easier. I choose this moment to acknowledge her contribution gratefully.***

***It is my radiant sentiment to place on record my best regards, deepest sense of gratitude to Mr. Mohammad Siraj and Mr. Siva, Sr. Software Developers, for their careful and precious guidance which were extremely valuable.***

***My thanks and appreciations also go to my teammates in developing the project and people who have willingly helped me out with their abilities.***

***I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future,***

***Sincerely,***

***Mohammad Anas***

# PROJECT REPORT

**PROJECT NAME**: MERIDIAN

**TASK ASSIGNED**: To develop a chat bot with a modernized user touchpoint and integrate it with the patient estimation API as well as the other API's of the meridian project, which internally connects with the back-end systems and provide the data as per the request.

**REPORT BY**:

MOHAMMAD ANAS

INTERN @ OPTUM GLOBAL SOLUTIONS

**TENURE**:3rd JUNE 2019 - 26th JULY 2019

**PRODUCT OWNER**: M.S SRINATH

**PROJECT MANAGER**:

SRUJANA SHREEVASTAV

**MENTOR**: SOURABH DWIVEDI

**H.R. MANAGER**:RAJNI

# CHATBOT

## WHAT IS A CHATBOT?

A chatbot (also known as a spy, conversational bot, chatterbot, interactive agent, conversational interface, Conversational AI, talkbot or artificial spy entity) is a computer program or an artificial intelligence which conducts a conversation via auditory or textual methods.

Such programs are often designed to convincingly simulate how a human would behave as a conversational partner, thereby passing the Turing test. Chatbots are typically used in dialog systems for various practical purposes including customer service or information acquisition.

Some chatbots use sophisticated natural language processing systems, but many simpler ones scan for keywords within the input, then pull a reply with the most matching keywords, or the most similar wording pattern, from a database.

## CHATBOT DEVELOPMENT TOOLS:

**1. Dialogflow**

Dialogflow is a conversational platform that lets you design and builds chatbots and voice apps (Google Actions and Amazon Alexa Skills). Dialogflow is backed by Google and is the most widely used tool to build Actions for more than 400M+ Google Assistant devices.

It supports all the major messaging channels such as Facebook Messenger, Slack, Skype, Kik, Line, Telegram, Twitter, Viber etc.

Dialogflow supports Natural Language Processing in 20+ languages.

Dialogflow is probably the best tools to build omnichannel chatbots with less coding. Dialogflow provides the REST API which can be used to integrate the chatbot to your own application or with custom conversational interfaces.

## 2. Microsoft Bot Framework

Microsoft Bot Framework connectors allow you to deploy chatbots on websites, apps, Cortana, Microsoft Teams, Skype, Slack, Facebook Messenger and more. It has 2 major components - Channel connectors and BotBuilder SDKs. Channel connectors allow you to connect the chatbot to messaging channels.

You can use the BotBuilder SDKs to implement the business logic in the chatbot. BotBuilder SDKs support C#, Javascript, Java, Python versions. BotBuilder comes with an Emulator for local debugging and visualization of conversations. This is really helpful for the developers during the development.

It's very easy for the developers to connect the Bot Builder SDK with any Natural Language Understanding (NLU) services. Bot Builder SDK Github account has many code samples and templates which help the developers to get started the chatbot development quickly.

## 3. Amazon Lex

Amazon Lex is a service for building conversational interfaces into any application using voice and text. Amazon Lex is the same technologies that power Amazon Alexa. As a fully managed service, Amazon Lex scales automatically, so you don’t need to worry about managing infrastructure.

With Amazon Lex, you can build, test, and deploy your chatbots directly from the Amazon Lex console itself.

Amazon Lex bots can be published to messaging platforms like Facebook Messenger, Slack, Kik, and Twilio SMS. Amazon Lex provides SDKs for iOS and Android for building bots for your mobile apps.

## 4. BotKit

Botkit is an opensource chatbot framework, recently acquired by Microsoft. Botkit is the leading developer tool for building chatbots, apps and custom integrations for major messaging platforms.

BotKit is NodeJs based SDK which supports publishing the chatbots on messaging channels such as Slack, Cisco Webex, Cisco Jabber, Microsoft Teams, Facebook Messenger Twilio SMS, Twilio IPM, Microsoft Bot Framework, Google Hangouts Chat.

Botkit also provides a web chat plugin which you can embed on any websites. With BotKit, you have to host the chatbot on your own server. BotKit can be easily used with all the major NLP platforms.

**5. BotPress**

Botpress is a dual-license open source bots development platform for developing bots in the same way that WordPress is a development platform for developing websites. The best part of Botpress is it provides UI where developers and non-technical people can manage the chatbots after the deployment.

Botpress has many nice features such as The Flow Builder and Dialog Manager. Flow Builder and Dialog Manager make it easier for developers to build and debug complex conversation flows. The developers can fully customize the chatbot - add business logic or integrate the 3rd party APIs etc.

Botpress is a good choice if your client wants to manage the chatbot contents by non-technical people after the deployment.

With Botpress you can deploy chatbot on Facebook, Slack, Telegram, BotFramework, Twilio, Web.

**6. Wit.ai**

Wit.ai is the Facebook acquired company. Wit is free, including for commercial use. It is an NLP platform which allows the developers to configure the entities and intents. Developers can use the HTTP API to connect the wit.ai to your chatbot or any other applications.

Wit.ai provides SDK in Node.js, Python, Ruby.

Wit.ai currently support Afrikaans, Albanian, Arabic, Azerbaijani, Bengali, Bosnian, Bulgarian, Burmese, Catalan, Central Khmer, Chinese, Croatian, Czech, Danish, Dutch, English, Estonian, Finnish, French, Georgian, German, Greek, Greenlandic, Hausa, Hebrew, Hindi, Hungarian, Icelandic, Igbo, Indonesian, Inuktitut, Italian, Japanese, Kannada, Kinyarwanda, Korean, Lao, Latin, Latvian, Lithuanian, Macedonian, Malay, Maori, Mongolian, Nepali, Norwegian, Pashto, Persian, Polish, Portuguese, Romanian, Russian, Serbian, Slovak, Slovenian, Somali, Southern Ndebele, Southern Sotho, Spanish, Swahili, Swati, Swedish, Tagalog, Tamil, Telugu, Thai, Tsonga, Tswana, Turkish, Ukrainian, Urdu, Uzbek, Venda, Vietnamese, Xhosa, Yoruba, and Zulu.

**7. Rasa Stack**

Rasa is an open source framework. It has two major components Rasa NLU and Rasa Core. Rasa NLU is responsible for natural language understanding. Rasa core is a framework for building conversational chatbot. Rasa core allows more sophisticated dialogue, trained using interactive and supervised machine learning.

The major advantage of using Rasa Stack is chatbot can be deployed on your own server by keeping all the components in-house. It is possible to use Rasa Core or Rasa NLU separately. Rasa is production ready and used in large companies everywhere.

Rasa core support Facebook Messenger, Rocket.Chat, Slack, Telegram, Twilio. Rasa is available under two license. Rasa NLU and Rasa Core are open sources. There is a paid and more advanced version of Rasa stack called Rasa platform.

Rasa Platform extends the open source Rasa NLU and Rasa Core libraries with APIs, a graphical user interface, and customer success program which includes enterprise-grade support. Rasa core is written in Python.

**8. IBM Watson Assistant**

Watson Assistant is an offering for building conversational interfaces into any application, device, or channel. You have the flexibility to deploy Watson Assistant on your site, in a mobile app, on the phone, in messaging channels, and to customer service tools.

It supports 13 languages. It provides SDK for the developers to build applications around Watson Assistant.

You can use SDKs in Java, Python, iOS. IBM offers free, standard, and premium plans.

# HOW TO BUILD A CHATBOT WITH DIALOG FLOW

## Chapter 1—Introduction

A chatbot is, in essence, a piece of robotic software used to imitate human conversation through text chats and voice commands (a good example being Siri or Amazon Alexa).

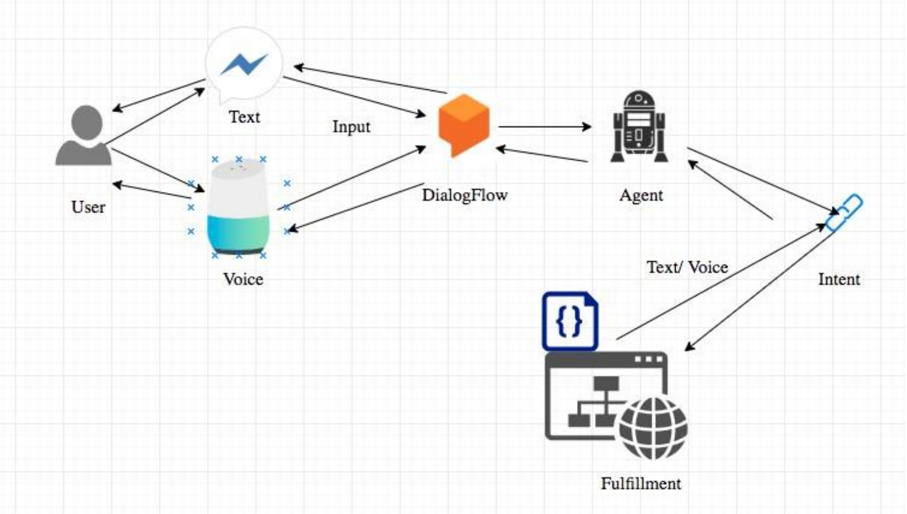
2 Types of chatbots:

1. Rule based chatbots (if you ask for phones the relevant phone pages open up in an e-commerce site that's an example of a rule based chatbot)
2. A.I. based chat bots (learn over a period of time using Machine Learning techniques)—dialog flow is an example of that

Chatbots are extremely valuable for businesses and this value will only increase as time goes by.

On obvious area of chatbot implementation is customer service. Bots are invaluable here. Waiting on hold may soon be a thing of the past as they become advanced enough to deal with basic level customer service queries, and this is already being used by a lot of companies worldwide. Nordstrom, for example, implemented a chatbot to assist with customer service at the end of 2016, and this has made their technical support much more responsive and immediate. It's no secret that this has resulted in significant cost reduction.

Text and Voice based chatbots are the future and if you are an entrepreneur or a techie, it's the right time to spend some time learning about building these bots.



# Flow of conversation within DialogFlow

**User**: We, Machines!

**Text / Voice** : The user interacts with an app like facebook messenger / google home to start the interaction with the bot.

**Dialogflow**: Bot platform

**Agent**: A module within dialogflow which incorporates Natural Language Processing to understand what the user meant and to figure out what "action" has to be carried out. The agent transforms the user request into machine readable actionable data.

**Intent:** Support or the service that the user wants from the agent. Intent is configured by the developers. Intent determines the action by the code.

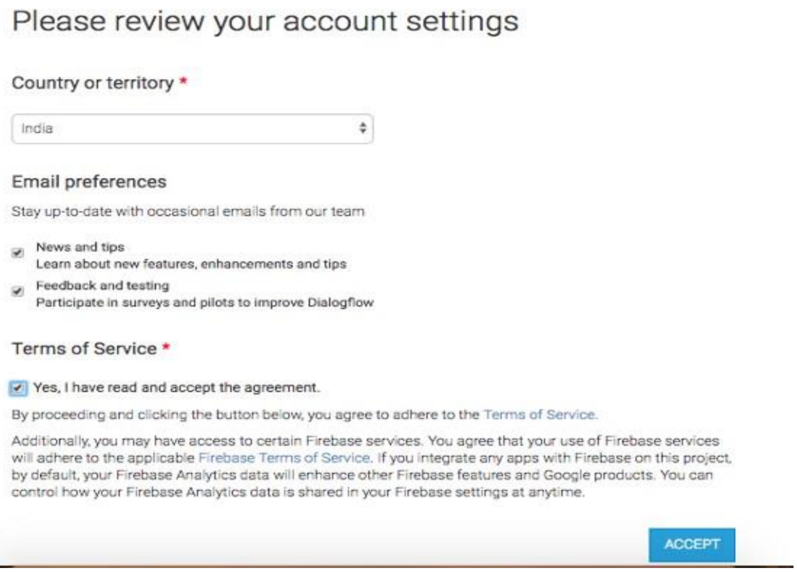
**Fulfillment:** This is the code. This part of the conversation lets you pass on the request from your bot to an external source and get response and pass it back to the user. This is achieved via Webhook. Setting up a webhook allows you to pass information from a matched intent into a web service and get a result from it.

Note: Don't be threatened by the terms here. Once we setup dialogflow account and open the site, all this will fall into place.

# Setting up DialogFlow account:

Ok now that the boring ‘theory / lecture’ part is over, let’s jump in and start setting up the environment where we’ll be creating our bot!

1. Goto : https://dialogflow.com/
2. Create an account with a gmail account, and “agree” to the terms & conditions.



Please review your account settings

Country or territory *

India

Email preferences

Stay up-to-date with occasional emails from our team

☑ News and tips
Learn about new features, enhancements and tips

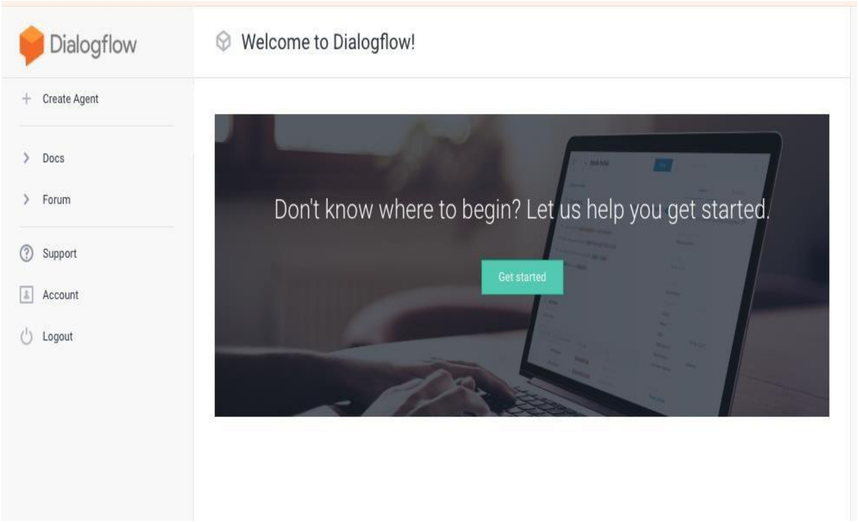
☑ Feedback and testing
Participate in surveys and pilots to improve Dialogflow

Terms of Service *

☑ Yes, I have read and accept the agreement.

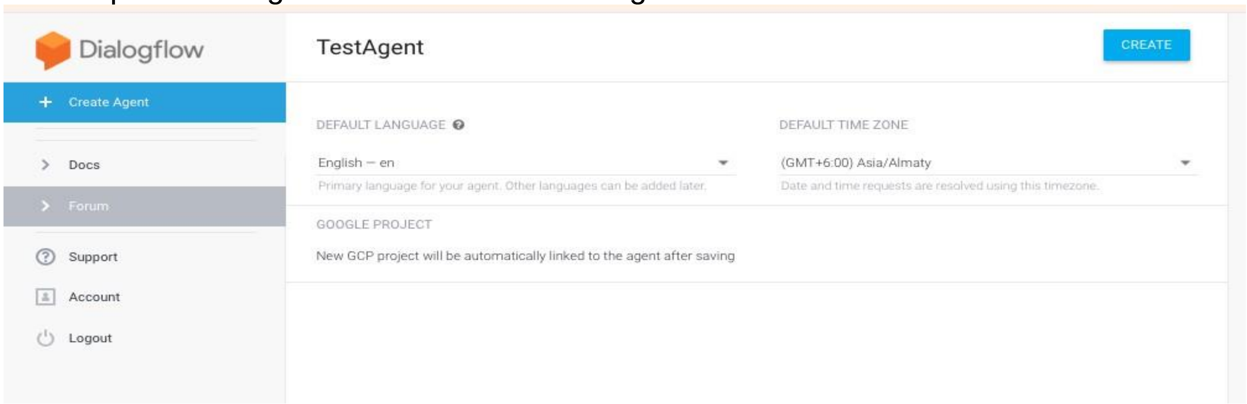
By proceeding and clicking the button below, you agree to adhere to the Terms of Service.

Additionally, you may have access to certain Firebase services. You agree that your use of Firebase services will adhere to the applicable Firebase Terms of Service. If you integrate any apps with Firebase on this project, by default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your Firebase settings at anytime.



# Authorise DialogFlow on Google cloud:

First step in creating our bot is to create an agent.



TestAgent

DEFAULT LANGUAGE

English — en

Primary language for your agent. Other languages can be added later.

DEFAULT TIME ZONE

(GMT+6:00) Asia/Almaty

Date and time requests are resolved using this timezone.

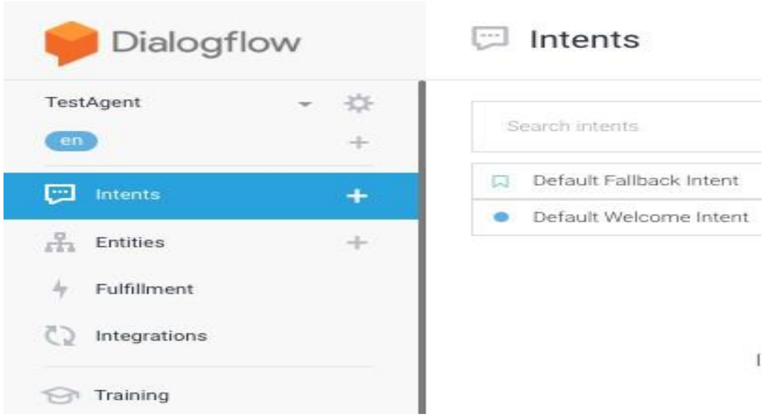
GOOGLE PROJECT

New GCP project will be automatically linked to the agent after saving

This will create a new GoogleCloud Project automatically. If you are prompted for the authorisation, do allow. If you don't have a google cloud platform account please create one.

# Chapter 2— Building Blocks of DialogFlow

Building blocks of Dialogflow.



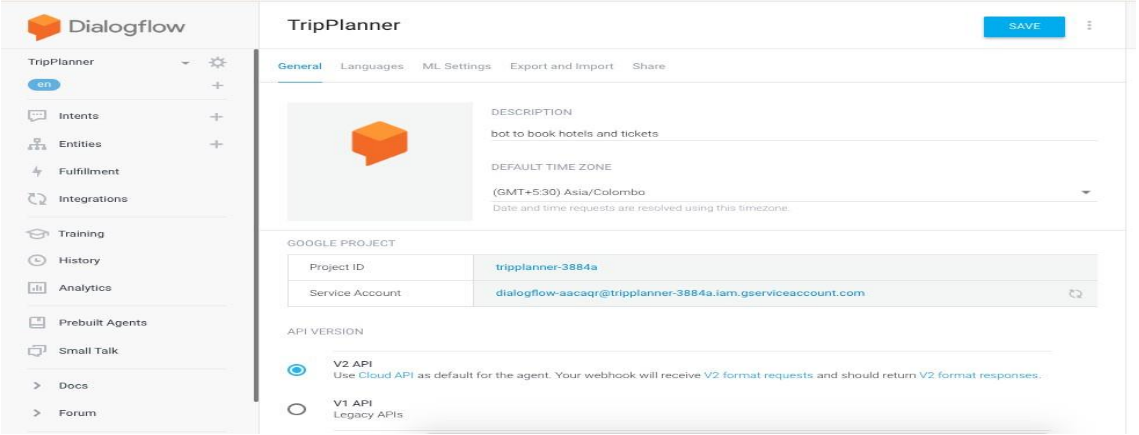
**Agents** : Help convert user request into actionable data. Eg: TestAgent in the image above

1. **Intents**: These are configured by developers which indicate what the objective of the user might be when he/she/she makes a specific request. Eg: Book a flight / Collect Feedback etc
2. **Entities**: Help extract information from user speech with the help of prompts. Eg: “Book a flight” intent might need such as the: to and from cities, date, class etc as entities that the agent tries to extract from the user via conversations. The information received here are sent on for fulfilment.
3. **Fulfilment**: Code that fulfils the intent of the user’s request.
4. **Integrations**: Twitter, Slack, Googlehome etc.
5. **Prebuilt Agents:** Check out the various built-in agents
6. **Smalltalk:** Helps make bots friendly and chatty with no coding from our end.

Agents:

Agents translate user requests to actionable data i.e. intents. It’s essentially a module within dialog flow which incorporates Natural Language Processing to understand what the user meant and to figure out what “action” has to be carried out. Agents manage conversations with the user through intent, entities, contexts and other building blocks. We are gonna build a bot that helps us with planning a business trip.

Build the Agent : Trip Planner



TripPlanner

DESCRIPTION
bot to book hotels and tickets

DEFAULT TIME ZONE
(GMT+5:30) Asia/Colombo
Date and time requests are resolved using this timezone.

GOOGLE PROJECT

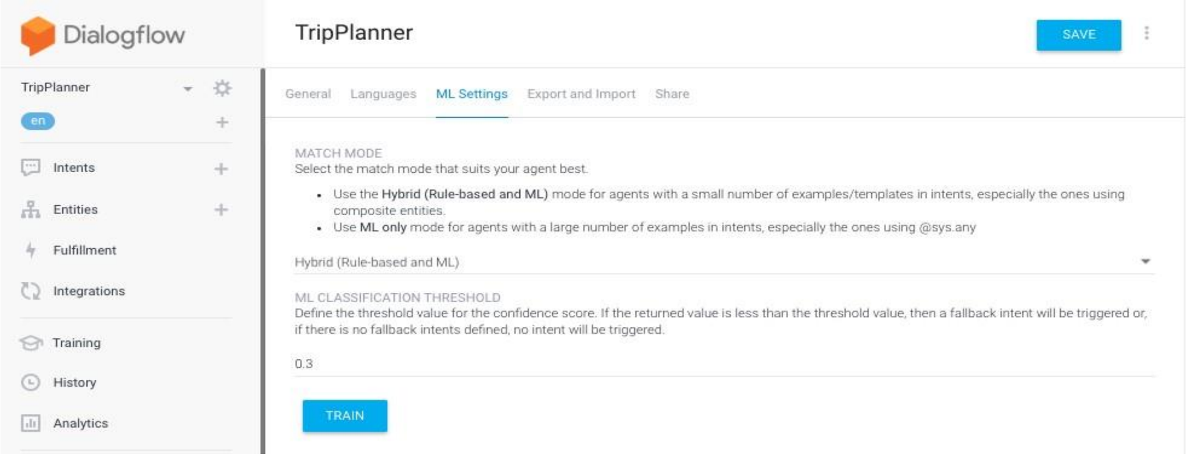
| | |
|---|---|
| Project ID | tripplanner-3884a |
| Service Account | dialogflow-aacaqr@tripplanner-3884a.iam.gserviceaccount.com |

API VERSION

● V2 API
Use Cloud API as default for the agent. Your webhook will receive V2 format requests and should return V2 format responses.

○ V1 API
Legacy APIs

Explore the settings page…the ML settings gives us the type of machine learning we wanna use for this bot.



MATCH MODE

Select the match mode that suits your agent best.

- Use the **Hybrid (Rule-based and ML)** mode for agents with a small number of examples/templates in intents, especially the ones using composite entities.
- Use **ML only** mode for agents with a large number of examples in intents, especially the ones using @sys.any

Hybrid (Rule-based and ML)

ML CLASSIFICATION THRESHOLD

Define the threshold value for the confidence score. If the returned value is less than the threshold value, then a fallback intent will be triggered or, if there is no fallback intents defined, no intent will be triggered.

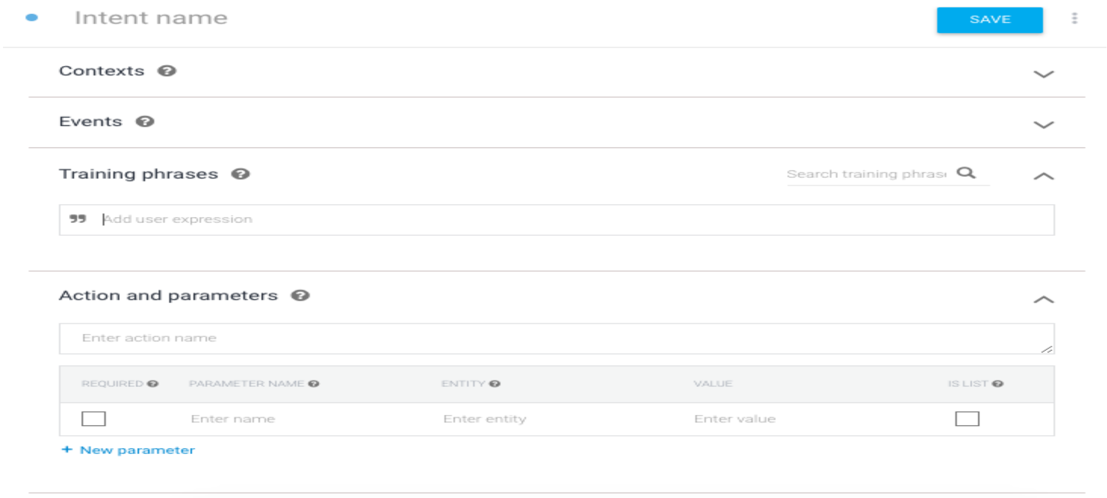
0.3

ML CLASSIFICATION THRESHOLD—0.3

Define the threshold value for the confidence score. If the returned value is less than the threshold value, then a fallback intent will be triggered or, if there is no fallback intents defined, no intent will be triggered. In simple terms, if the match is less than 30% percent then the inbuilt “fallback intent” (sorry, I don’t understand etc) will be triggered. Try to go through the settings tab and the google cloud platform project that has been created automatically the moment you created the agent.

Intent:

Intents are configured by developers and used to determine the action taken by the code. Think of using Intent as a mapping….what a user says and what your software should execute. <Don’t rush to create these intents…am just explaining stuff here, once we reach custom intent, will show how these intents are created>

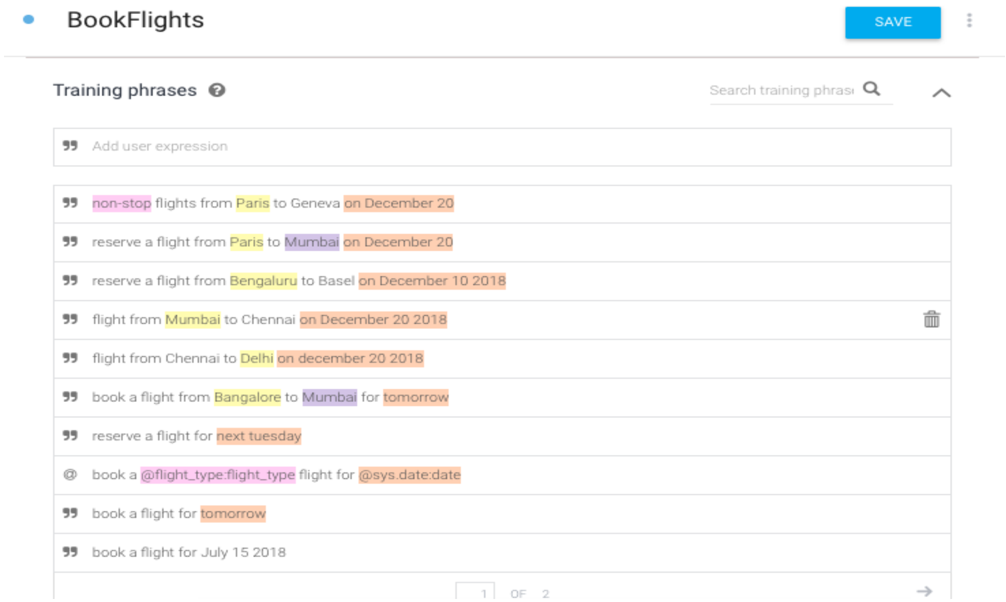


| REQUIRED | PARAMETER NAME | ENTITY | VALUE | IS LIST |
|---|---|---|---|---|
| □ | Enter name | Enter entity | Enter value | □ |

Don’t worry about context and events for now. We’ll get there in chapter 3.

**Training Phrase:**

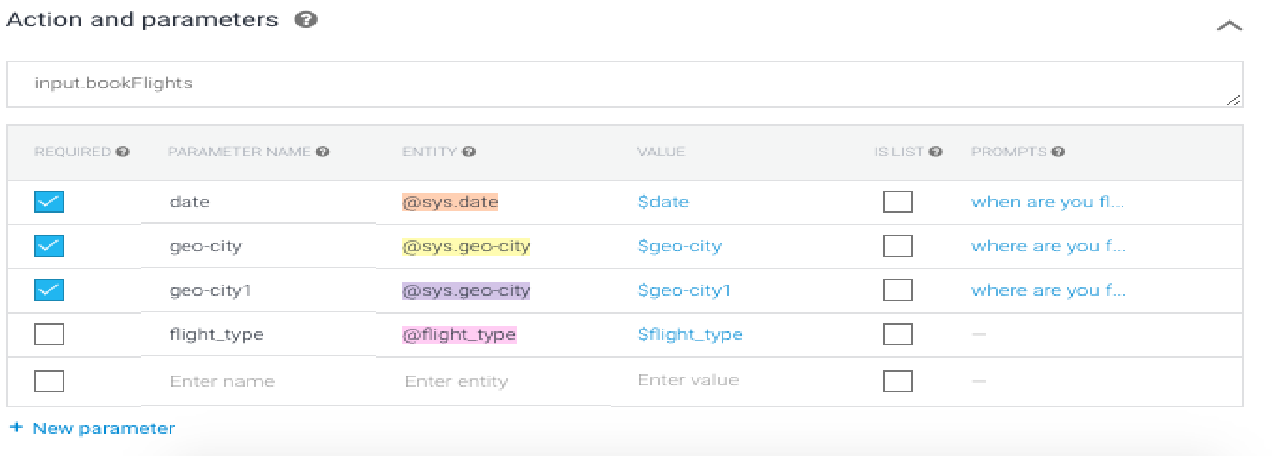
Phrases you can expect from the user that will trigger the intent.



- non-stop flights from Paris to Geneva on December 20
- reserve a flight from Paris to Mumbai on December 20
- reserve a flight from Bengaluru to Basel on December 10 2018
- flight from Mumbai to Chennai on December 20 2018
- flight from Chennai to Delhi on december 20 2018
- book a flight from Bangalore to Mumbai for tomorrow
- reserve a flight for next tuesday
- book a @flight_type:flight_type flight for @sys.date:date
- book a flight for tomorrow
- book a flight for July 15 2018

**Action & Parameters:**

The inputs you might need from the user to take an action on the user request. For eg: In BookFlights intent, we might need the To city, From City, Date etc to finish the action. We have to set entities first for it to be mapped here but go with the flow for now.



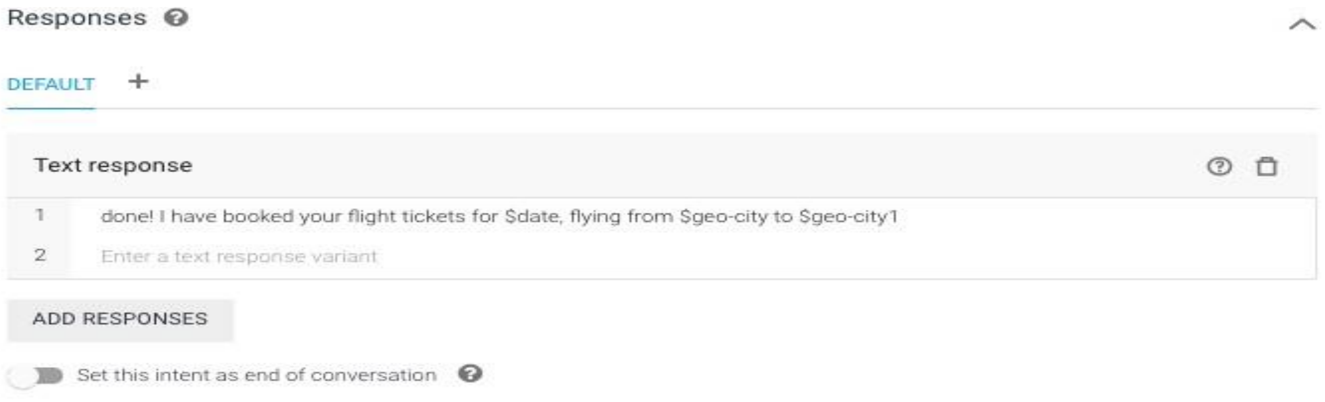
Action and parameters

input.bookFlights

| REQUIRED | PARAMETER NAME | ENTITY | VALUE | IS LIST | PROMPTS |
|---|---|---|---|---|---|
| ☑ | date | @sys.date | $date | □ | when are you fl... |
| ☑ | geo-city | @sys.geo-city | $geo-city | □ | where are you f... |
| ☑ | geo-city1 | @sys.geo-city | $geo-city1 | □ | where are you f... |
| □ | flight_type | @flight_type | $flight_type | □ | — |
| □ | Enter name | Enter entity | Enter value | □ | — |

+ New parameter

**Response:**



Responses

Text response

1 done! I have booked your flight tickets for $date, flying from $geo-city to $geo-city1

2 Enter a text response variant

Set this intent as end of conversation

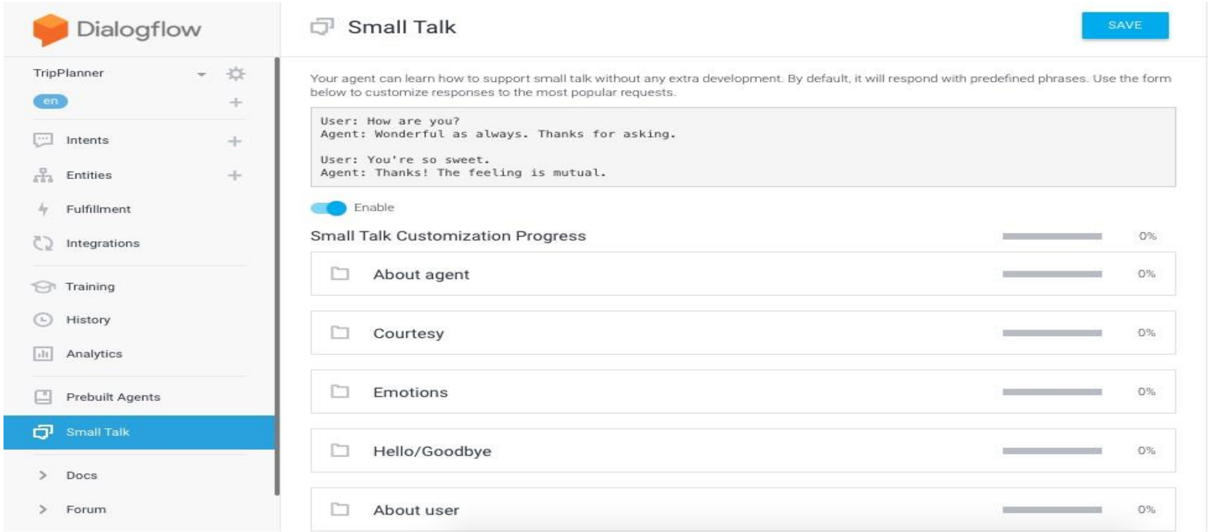
## Default Intents:

Dialogflow has few default intents that can help us save a lot of time.

1. **Welcome Intent**: Greets the usr, exchange pleasantries (partially configured)
2. **Fallback Intent** : Default fall through intent when no others match (excuse me, I din't get what you said)

   Small Talk Intent:

Helps make bots friendly and chatty with no coding from our end. Again, this comes by default you just have to enable it.



Your agent can learn how to support small talk without any extra development. By default, it will respond with predefined phrases. Use the form below to customize responses to the most popular requests.

```
User: How are you?
Agent: Wonderful as always. Thanks for asking.

User: You're so sweet.
Agent: Thanks! The feeling is mutual.
```

Enable

| Small Talk Customization Progress | 0% |
|---|---|
| About agent | 0% |
| Courtesy | 0% |
| Emotions | 0% |
| Hello/Goodbye | 0% |
| About user | 0% |

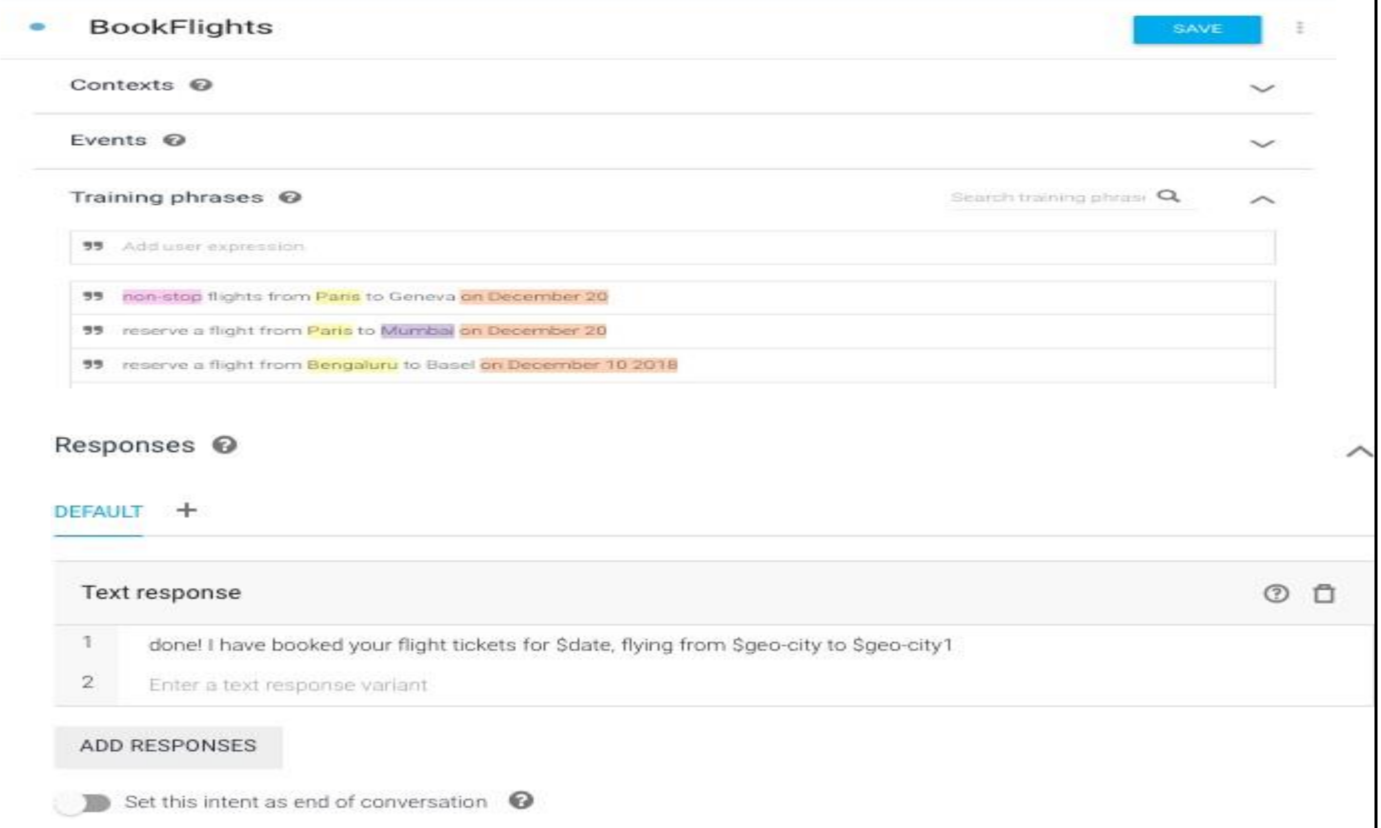
**Custom Intent:**

Continuing our effort to create the TripPlanner bot...we gotta create the intents for the same. We are gonna help users book a flight, book a room and book a car so we need to create 3 custom intents

*create custom intents*

So we gotta create 3 custom intents.

1. BookFlights



BookFlights

Training phrases

non-stop flights from Paris to Geneva on December 20

reserve a flight from Paris to Mumbai on December 20

reserve a flight from Bengaluru to Basel on December 10 2018

Responses

Text response

1 done! I have booked your flight tickets for $date, flying from $geo-city to $geo-city1

2 Enter a text response variant

Set this intent as end of conversation

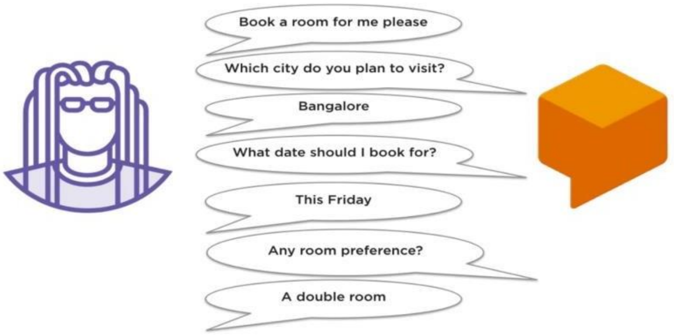
if the cities and the date aren’t getting tagged / colored don’t panic..it’ll happen after we create the entities.

Similarly, create two more custom intents …. BookCars, BookRooms

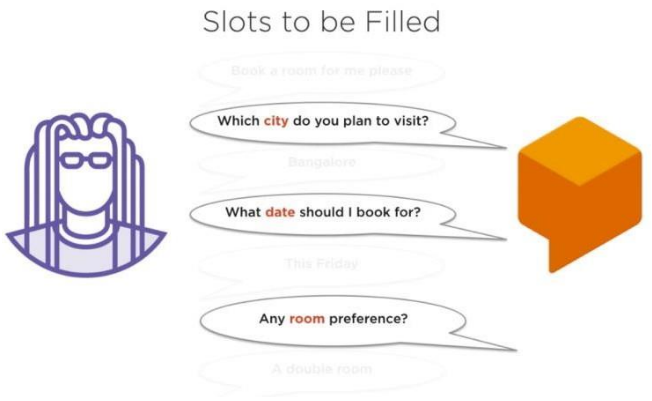
Entities:

So, till now we have created one agent and three custom intents. Let’s understand entities in this section.

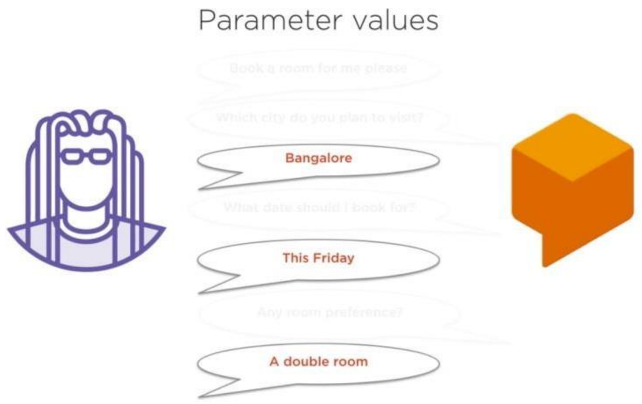
Entities are used to extract parameter values from user queries. So when the user says “Book me a flight” we usually ask for the city, date and the type of flight probably..these are called entities.


Book a room for me please
Which city do you plan to visit?
Bangalore
What date should I book for?
This Friday
Any room preference?
A double room

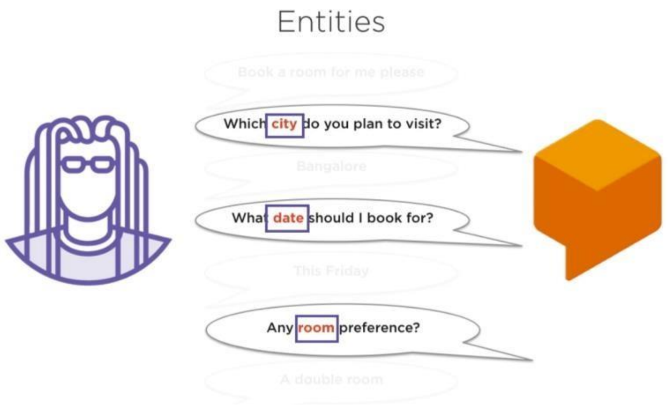
**Example Conversation**


Slots to be Filled
Book a room for me please
Which city do you plan to visit?
Bangalore
What date should I book for?
This Friday
Any room preference?
A double room

**We (chatbot) asks questions to fill our Slots**


Parameter values
Book a room for me please
Which city do you plan to visit?
Bangalore
What date should I book for?
This Friday
Any room preference?
A double room

**Whatever the user responds to the slot based questions are the parameters**


Entities
Book a room for me please
Which city do you plan to visit?
Bangalore
What date should I book for?
This Friday
Any room preference?
A double room

**The generic abstract form for the parameters are the entities**

**Three types of entities:**

- System—Time, City, Date
- Developer—Room type
- User—defined at the session level (user’s playlist)..think of them as cookies in websites

Ok, time to work:

*Create Developer Entities:*

Book Flights Intent:

1. From City
2. To City
3. Date
4. Flight Type [optional]

Book Rooms Intent:

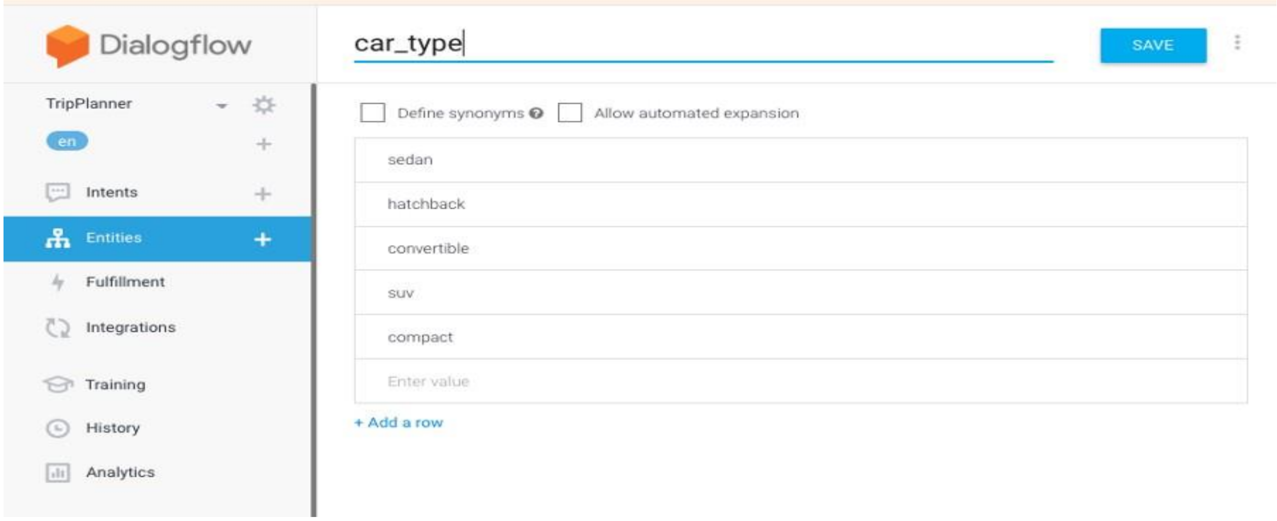
1. City
2. Date

3. Room Type [Optional]

Book Cars Intent:

1. City
2. Date
3. Car Type

So as you can see City and Date are system entities so we don't have to define them..but flight_type, room_type, car_type are all custom entities and we gotta define them..see images below.

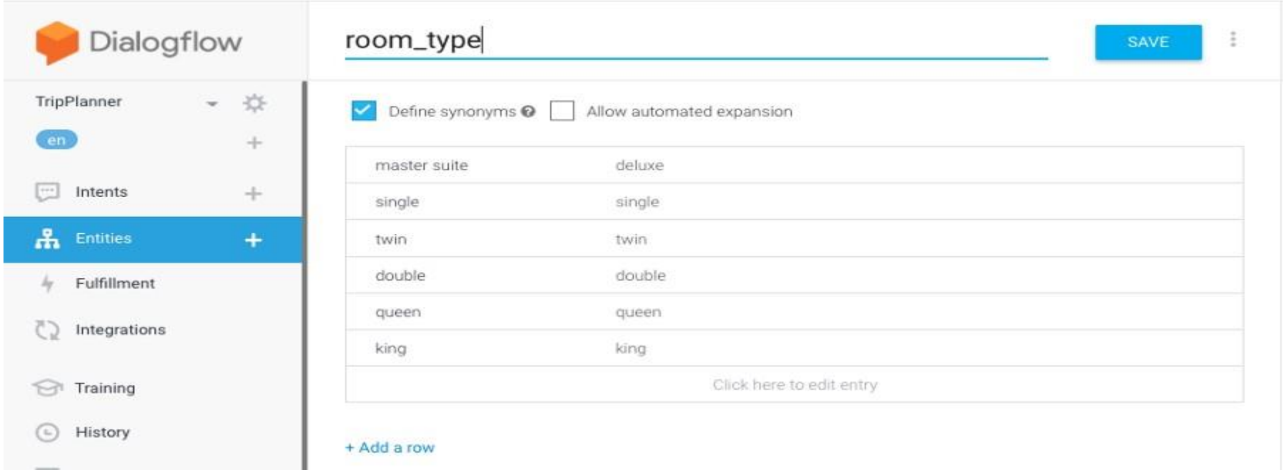


car_type

☐ Define synonyms ☐ Allow automated expansion

sedan
hatchback
convertible
suv
compact
Enter value

+ Add a row



room_type

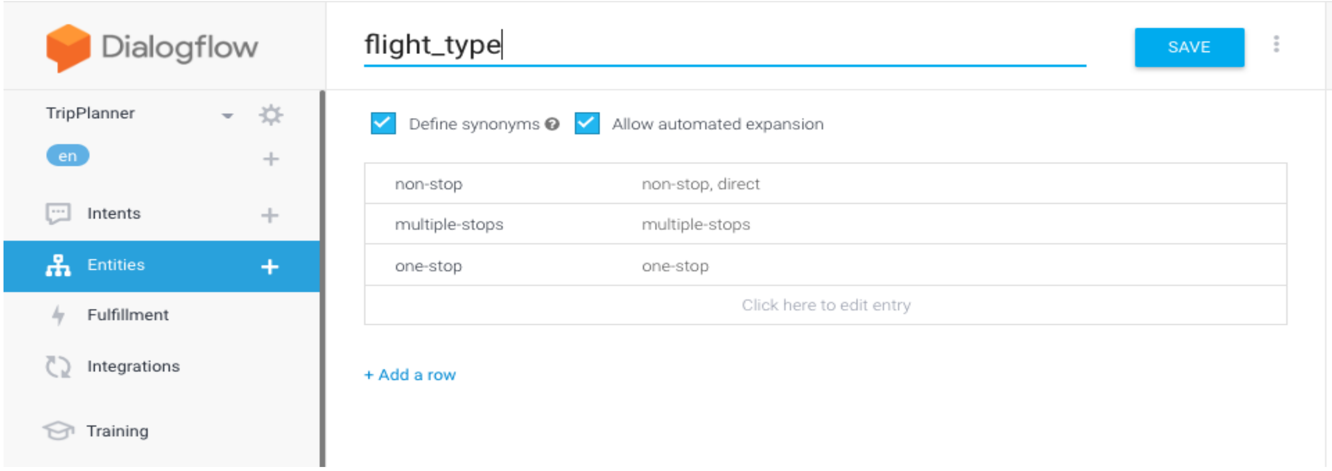
☑ Define synonyms ☐ Allow automated expansion

| | |
|---|---|
| master suite | deluxe |
| single | single |
| twin | twin |
| double | double |
| queen | queen |
| king | king |

Click here to edit entry

+ Add a row

custom entities are configured



flight_type

☑ Define synonyms ☑ Allow automated expansion

| | |
|---|---|
| non-stop | non-stop, direct |
| multiple-stops | multiple-stops |
| one-stop | one-stop |

Click here to edit entry

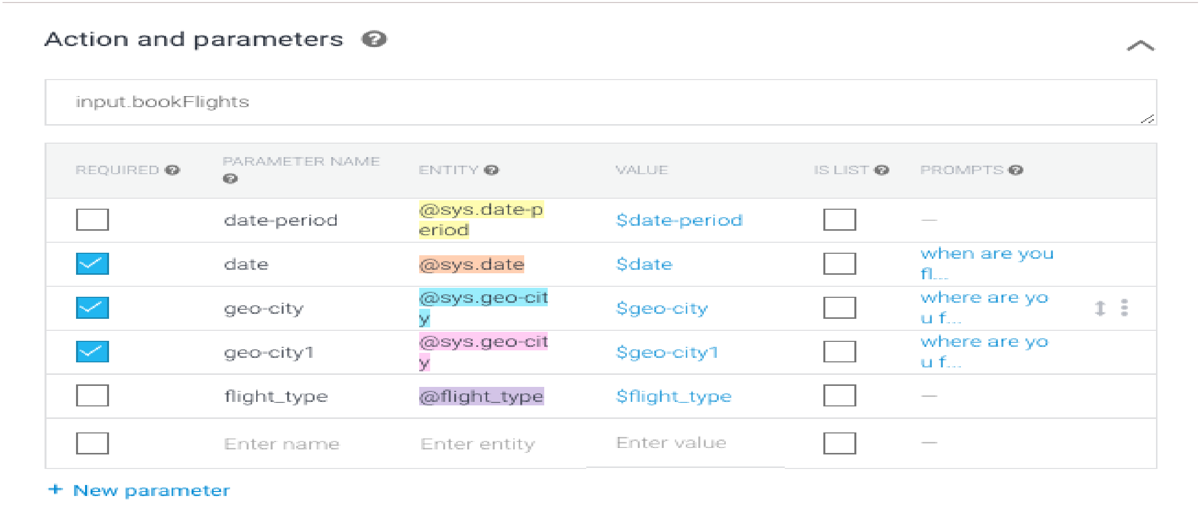
+ Add a row

Now that the Entities are defined lets go back and configure the intents we had created earlier.

Configuring Custom Intent

Book Flight intent with

- Expression with annotations
- Parameters
- Prompts

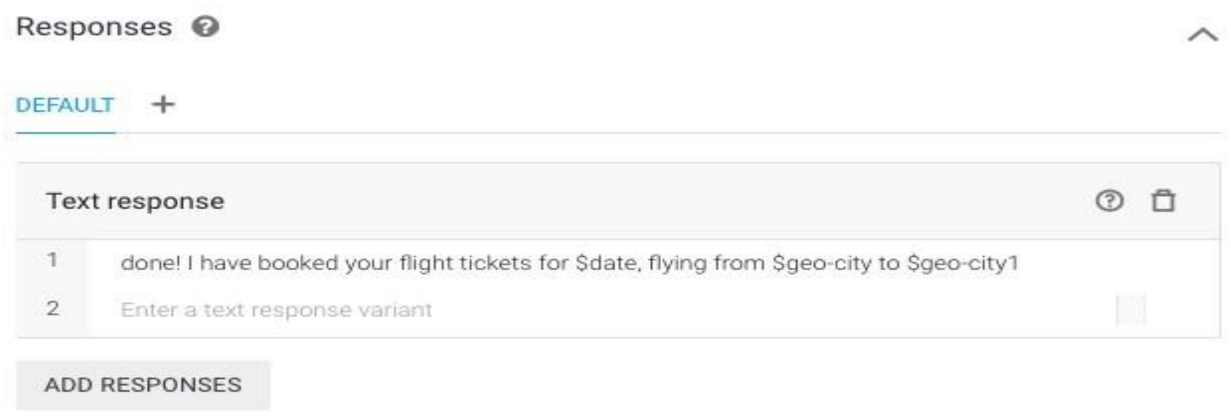


Action and parameters

input.bookFlights

| REQUIRED | PARAMETER NAME | ENTITY | VALUE | IS LIST | PROMPTS |
|---|---|---|---|---|---|
| □ | date-period | @sys.date-period | $date-period | □ | — |
| ☑ | date | @sys.date | $date | □ | when are you fl... |
| ☑ | geo-city | @sys.geo-city | $geo-city | □ | where are you f... |
| ☑ | geo-city1 | @sys.geo-city | $geo-city1 | □ | where are you f... |
| □ | flight_type | @flight_type | $flight_type | □ | — |
| □ | Enter name | Enter entity | Enter value | □ | — |

+ New parameter



Responses

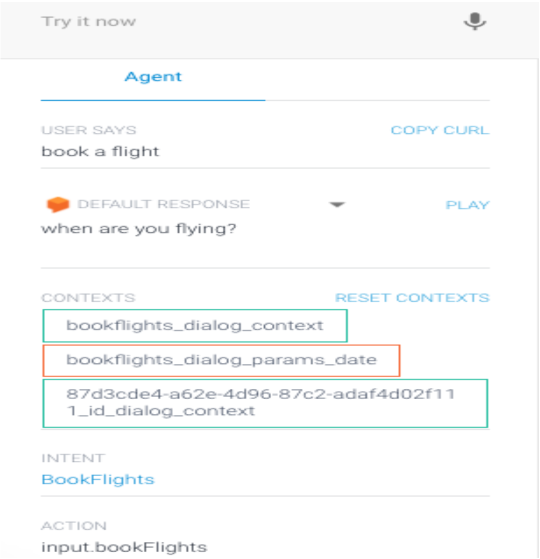
DEFAULT +

Text response

1 done! I have booked your flight tickets for $date, flying from $geo-city to $geo-city1

2 Enter a text response variant

ADD RESPONSES



Try it now

Agent

USER SAYS COPY CURL

book a flight

DEFAULT RESPONSE PLAY

when are you flying?

CONTEXTS RESET CONTEXTS

bookflights_dialog_context

bookflights_dialog_params_date

87d3cde4-a62e-4d96-87c2-adaf4d02f111_id_dialog_context

INTENT

BookFlights

ACTION

input.bookFlights

# HOW TO BUILD A CHATBOT WITH RASA STACK & RASA CORE

{Optum supported platform}

## Step One: Prerequisites

Before getting started with Optum ChatBot, you need to request secure access to 'bots_ai_sandbox' MS group.

1. Navigate to Secure
2. Under **Primary Windows Account,** click on **Add Group Membership**
3. From the **Groups** section, enter the following in the search box "bots_ai_sandbox"
4. Select "bots_ai_sandbox"
5. Click on the green arrow
6. Click the **next button**
7. Enter "Optum Developer Chatbot Getting Started" and submit the **request**
8. Once your request has been approved, proceed to the next step

## Step Two: Create your first Chatbot

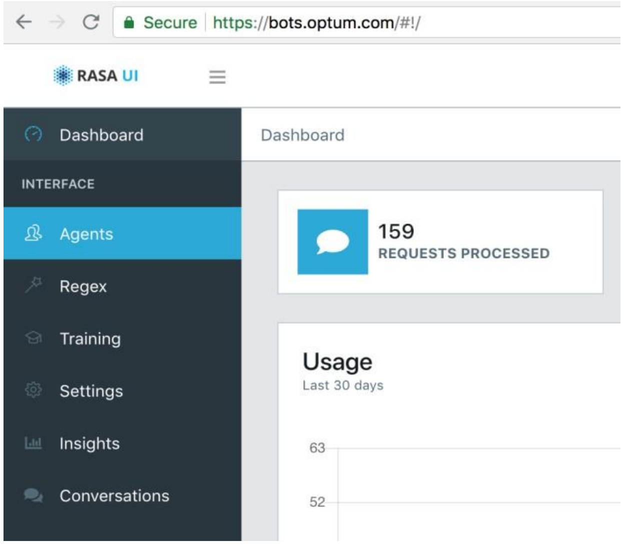
Once you've met the prerequisites, follow the next steps to get started with creating your first Chatbot.

- Go to https://bots.optum.com/ and login with your MSId

- Follow the steps below

### Step One: Create an Agent

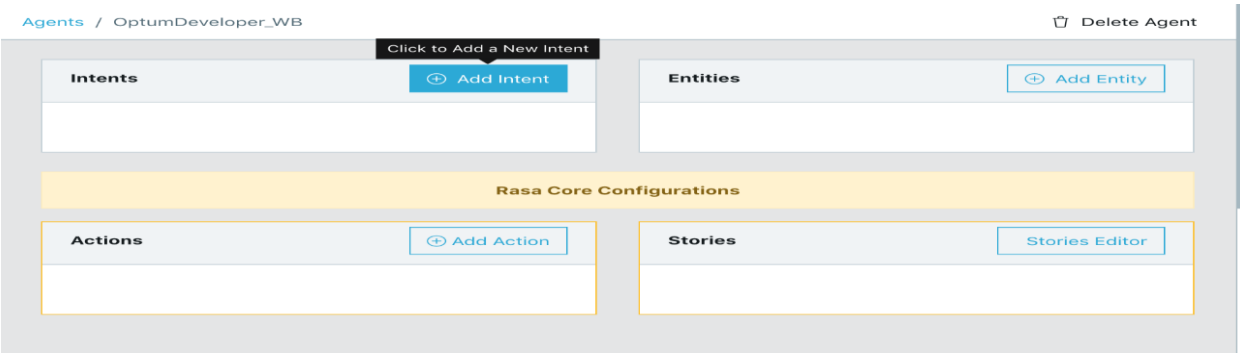
- On the home page, navigate to the Agents section by clicking on **Agents** in the left side navigation menu
- Create a new agent by clicking on **Add Agents** at the top right corner of the center screen
  - Enter your new agent's name
  - Name: *OptumDevBot_YourNameHere*
  - Press Enter or Click the Save button
- Access your newly created agent
  - In the agents page, locate your new agent by navigating to the last available agents page
  -
  - *Note: You can flip between pages by clicking on the page numbers or left and right arrows in the bottom right of the agents page.*
  - Once you've found your agent, click on your agent's name.



### Step Two: Add a new Intent

- Add a new 'greet' intent for your agent
  - Under the intents section in the top left box, click on **Add Intent**
  - Enter an intent name

- ○ Name: *greet_intent*
- ○ Press Enter or Click the Save button



**Step Three: Add Expressions to an Intent**

Add expressions for your 'greet' intent

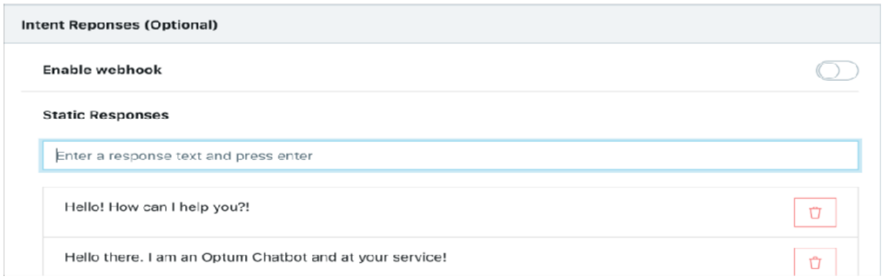
- In your agent's page, click on your newly created intent
- Under **Expressions** and **User Say's...,** input form add the two following expressions and pressing Enter

1. hi
2. hello

**Step Four: Add Static Responses to an Intent**

Add static responses for your 'greet' intent

- In your agent's 'greet' intent page, click on the input form below **Static Responses** in the **Intent Responses** section
- Add the following three responses
  - ○ Hello! How can I help you?!
  - ○ Hello there. I am an Optum Chatbot and at your service!
  - ○ Hi! I am Optum Chatbot. Say 'Help' for more information on how I can help you!
- Click the save button


Intent Reponses (Optional)
Enable webhook
Static Responses
Enter a response text and press enter

Hello! How can I help you?!

Hello there. I am an Optum Chatbot and at your service!

**Step Five: Add a Second Intent**

□ Add a new 'help' intent for your agent

□ Add expressions for your 'help' intent

o In your agent's page, click on your newly created 'help' intent

o Under **Expressions** and **User Say's...,** input form add the two following expressions and pressing Enter

1. help
2. help me

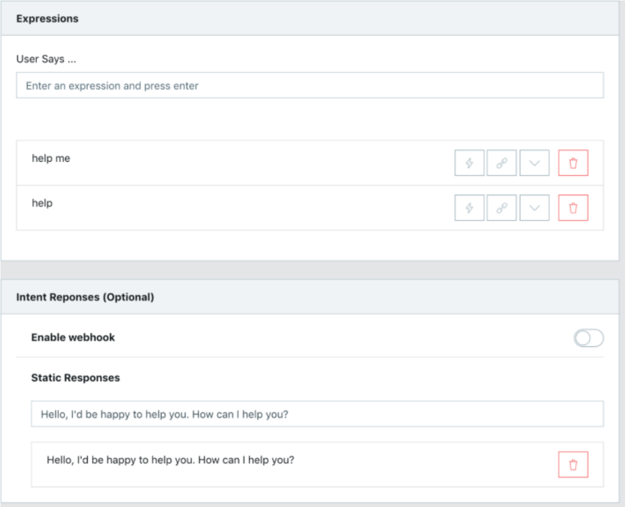
□ Add static responses for your 'help' intent

o In your agent's 'help' intent page, click on the input form below **Static Responses** in the **Intent Responses** section

o Add the following response

1. Hello, I'd be happy to help you. How can I help you?

o Click the Save button



Expressions

User Says ...

Enter an expression and press enter

help me

help

Intent Reponses (Optional)

Enable webhook

Static Responses

Hello, I'd be happy to help you. How can I help you?

Hello, I'd be happy to help you. How can I help you?

# Step Three: Train your Chatbot

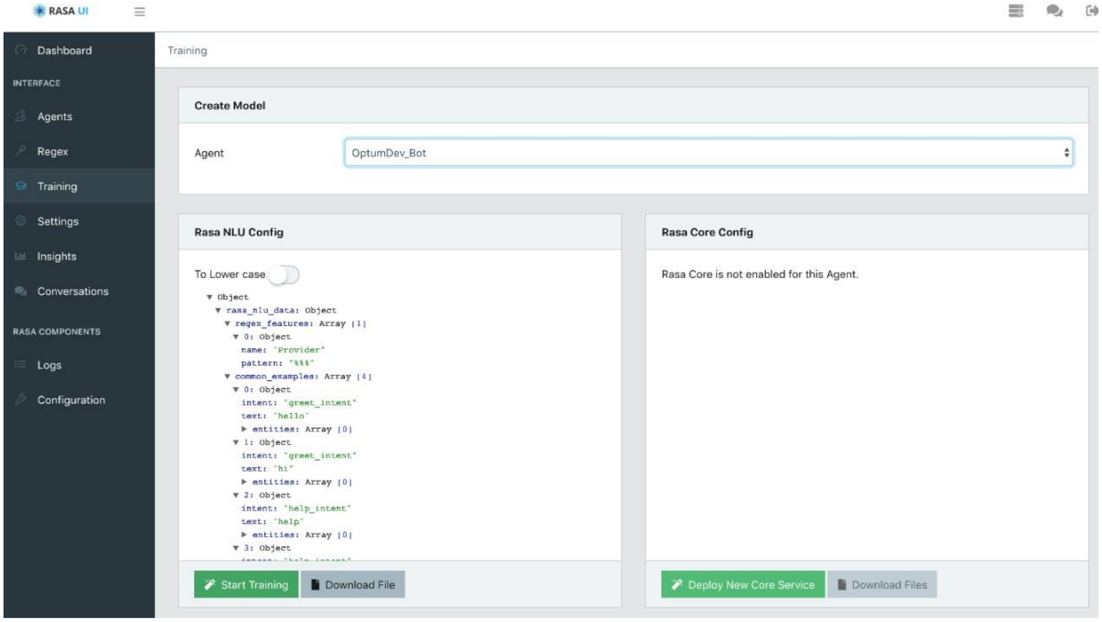
Once you've created your first Chatbot with intents and responses, you can now train your bot. A new chatbot model will be generated after the training process is completed. This model will allow you to test your bot. Let's find out how to train your Chatbot!

□ Click on **Training** in the left side navigation menu

□ Under **Create Model** section, click on the **Agent** drop down menu and select your Agent

□ The **Rasa NLU Config** section will then appear on your screen. Click on the green **Start Training** button below to begin training your bot.

**Note:** During bot training, a blue notification may appear above to show that training has begun. Once the training is completed, a green notification will appear. You're now ready to move on to the next step!



Agent: OptumDev_Bot

To Lower case

```
▼ Object
  ▼ rasa_nlu_data: Object
    ▼ regex_features: Array [1]
      ▼ 0: Object
        name: "Provider"
        pattern: "%%%"
    ▼ common_examples: Array [4]
      ▼ 0: Object
        intent: "greet_intent"
        text: "hello"
        ▶ entities: Array [0]
      ▼ 1: Object
        intent: "greet_intent"
        text: "hi"
        ▶ entities: Array [0]
      ▼ 2: Object
        intent: "help_intent"
        text: "help"
        ▶ entities: Array [0]
      ▼ 3: Object
        intent: "help_intent"
```

Rasa Core is not enabled for this Agent.

# Step Four: Test your Chatbot

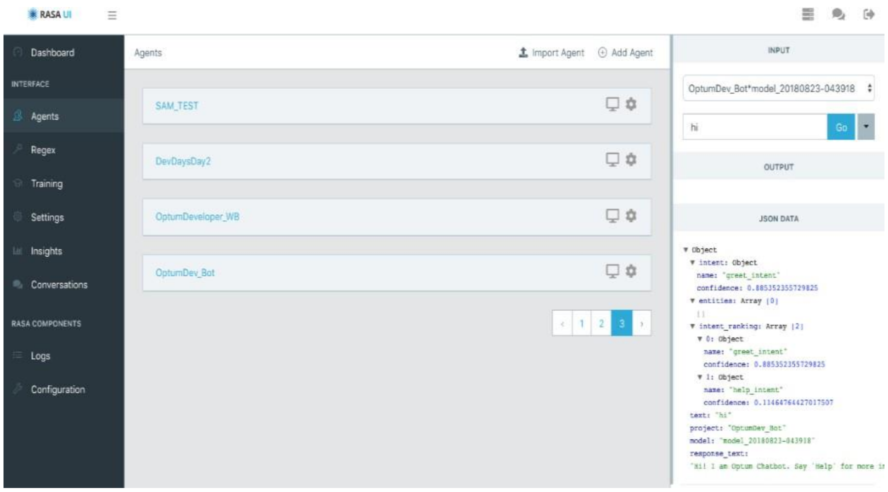
Now that your Chatbot has been trained, let's test it. There are two ways to demo your Chatbot. Follow the steps below to find out how to demo your bot!

**Step One: Console Demo**

Rasa UI Console Demo

- Note: the Rasa UI Console demo is a simple way to quickly test your bot. It will only return a JSON formatted response.
- Navigate to the **Agents** section
- the console demo will appear on the right side of the screen
- Under the drop down below **Input**, select your newly created model.
  - Note: Your Model will often have your Agent's name and the timestamp of when the model was created
- Enter some text input such as "help" or "hi"
- The trained model will take your input and determine which is the most likely intent and provide a static response we previously added.
- Try other phrases for greetings and saying help and observe the results

- Note: If the bot has failed to recognize your greet or help intent, you will likely need to add more expressions to your intent and retrain your bot

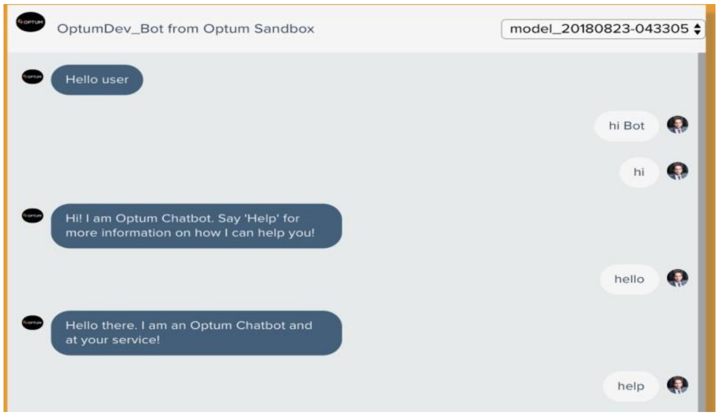


```
▼ Object
  ▼ intent: Object
    name: "greet_intent"
    confidence: 0.885352355729825
  ▼ entities: Array [0]
    []
  ▼ intent_ranking: Array [2]
    ▼ 0: Object
      name: "greet_intent"
      confidence: 0.885352355729825
    ▼ 1: Object
      name: "help_intent"
      confidence: 0.11464764427017507
  text: "hi"
  project: "OptumDev_Bot"
  model: "model_20180823-043918"
  response_text:
  "Hi! I am Optum Chatbot. Say 'Help' for more i
```

**Step Two: Live Demo**

Live Demo

- Note: Rasa UI also offers a way to demo your Chatbot in a live demo environment. This environment is a lot more similar to what an actual end user may experience
- Navigate to the **Agents** section and find your Agent
- On the same row of where your agent is located, click on the "monitor" icon to launch a live demo session
- Confirm that your trained model is selected in the dropdown of the top right hand corner
- Note: It may take some time for the Chatbot to started when live demoing for the first time
- Enter some text input such as "help" or "hi"



Hello user

hi Bot

hi

Hi! I am Optum Chatbot. Say 'Help' for more information on how I can help you!

hello

Hello there. I am an Optum Chatbot and at your service!

help

You've reached the end of these instructions for testing your first chatbot. Click on the next button to proceed to the next step and find out how to create a more advanced chatbot.

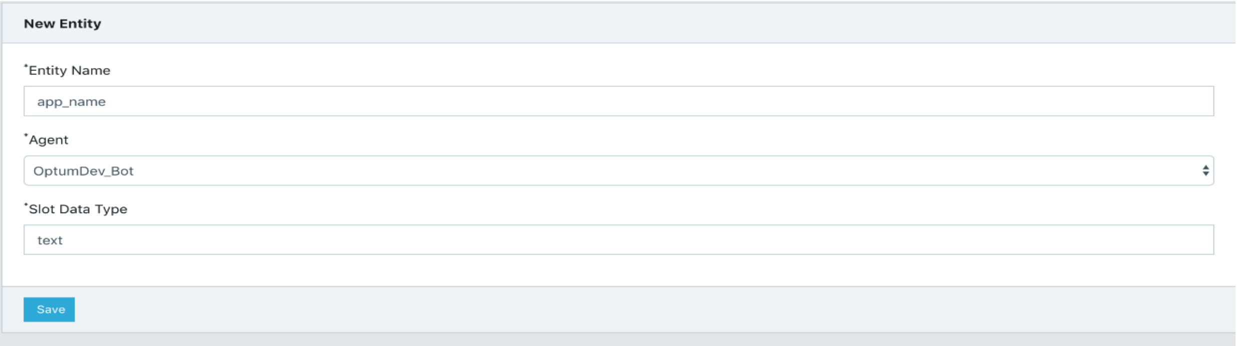
# Step Five: Add an Entity & Webhook

Entities allow us to identify key parameters for your API from a given user's input. For example, "What's the forecast for Nashville?", may have an entity called "city". Let's learn how to add entities and use them in an intent!

**Add an Entity**

In this section we're going to setup an entity for an API which provides information on an application's infrastructure cost. Let's assume our API accepts two parameters, an App Name and Infrastructure Type.

- Navigate to your Agent's menu page and select 'Add Entity'
- enter "app_name" under the **Entity Name** input field
- select "text" under the **Slot Data Type** selector.
- click Save



New Entity

*Entity Name: app_name

*Agent: OptumDev_Bot

*Slot Data Type: text

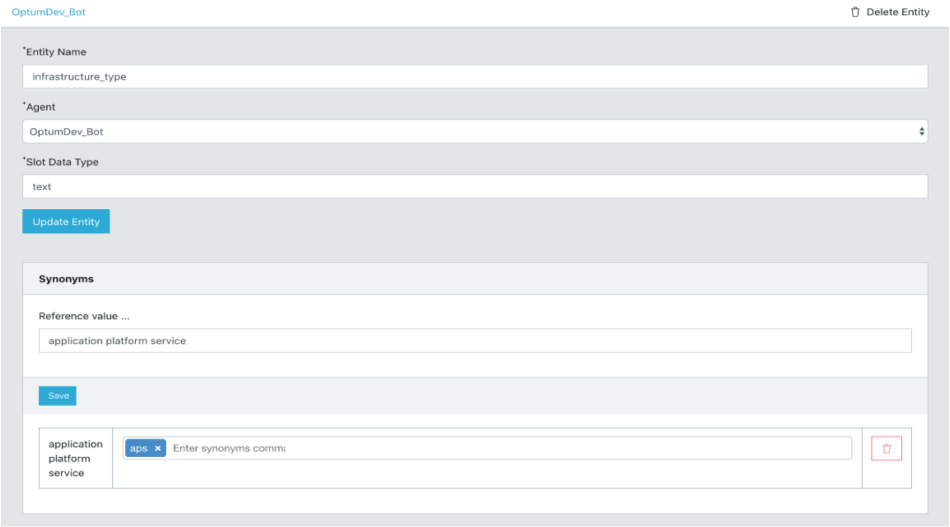
Nice work! Now we're going to setup the second entity which will be for the "infrastructure type" parameter in our API. However, we're anticipating that users may use different synonyms for a given acceptable value in our API. How do we account for synonyms?

For example, let's assume "application platform service" is an acceptable value for the "infrastructure type" parameter, but a user may only provide "aps" as a value. Follow the steps below to see how we can resolve this issue.

- Create a new entity with the **Entity Name** as "infrastructure_type"
- Set the **Slot Data Type** as "text"

- Under the **Synonyms** section and **Reference value** input field, enter "application platform service" and click save
- On the column next to **application platform service**, enter "aps"
- Click on Update Entity

Great! Now whenever our chatbot identifies an entity in a user's input as "infrastructure_type" with the value "aps", it will automatically set it to "application platform service". We can continue to do this for additional values that have multiple synonyms depending on our API's requirements.



OptumDev_Bot

Delete Entity

*Entity Name

infrastructure_type

*Agent

OptumDev_Bot

*Slot Data Type

text

Update Entity

Synonyms

Reference value ...

application platform service

Save

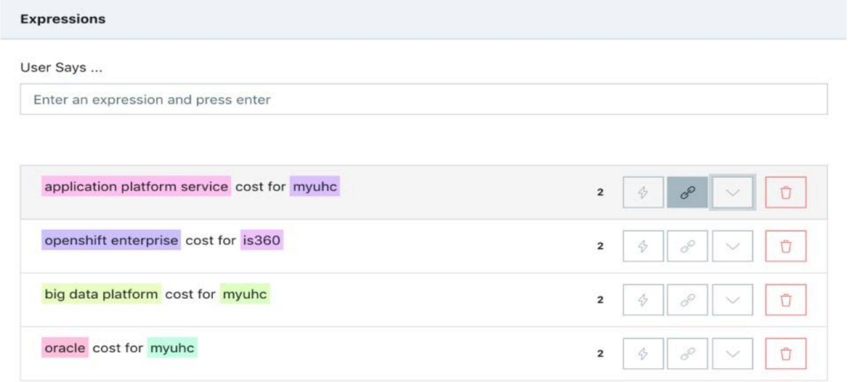
| application platform service | aps × Enter synonyms comm |
|---|---|

**Create an Intent with Entities**

- Create a new intent called "cost_query" and edit it
- With our new intent, we are now going to provide some training data and mark our entities within each given expression.
- Add the 4 following training data expressions below

1. *application platform service cost for myuhc*
2. *openshift enterprise cost for is360*
3. *big data platform cost for myuhc*
4. *oracle cost for myuhc*

- Mark our entity values in the expression by highlighting the word(s) for a given entity and then click on the chain icon. The entity in our expression should now be highlighted and you should see the total entity count on the right increment.
- Repeat the last step for each of expressions as shown in the next page



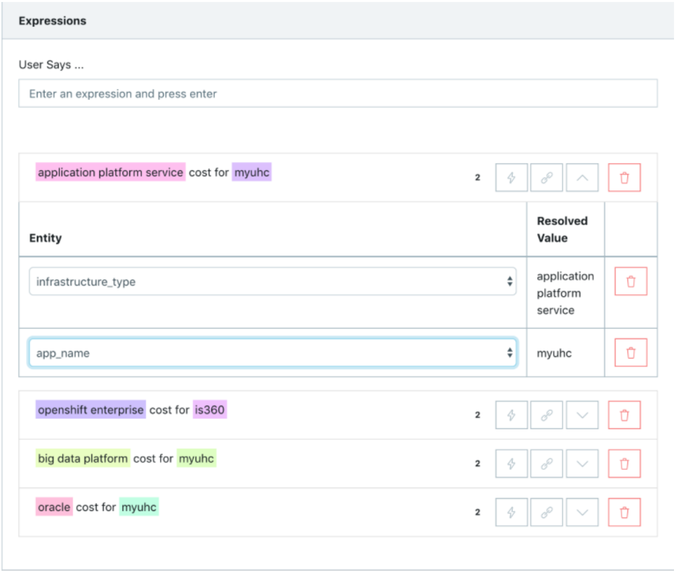
Expressions

User Says ...

Enter an expression and press enter

| | |
|---|---|
| application platform service cost for myuhc | 2 |
| openshift enterprise cost for is360 | 2 |
| big data platform cost for myuhc | 2 |
| oracle cost for myuhc | 2 |

- Assign these entity values to an entity we created earlier
- Click on the down arrow icon to open the entity-value assignment section
- For each entity value, click on the selector to its left and select the correct entity value.
  - Note: "Myuhc" and "Is360" should be assigned to app_name entity
- Assign the entity values to its correct entity for all 4 expressions
- Click save to save your work



Expressions

User Says ...

Enter an expression and press enter

application platform service cost for myuhc 2

| Entity | Resolved Value |
|---|---|
| infrastructure_type | application platform service |
| app_name | myuhc |

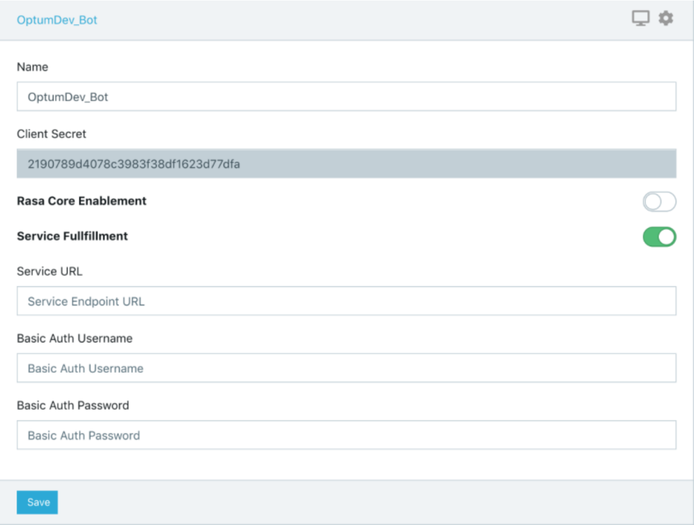
| | |
|---|---|
| openshift enterprise cost for is360 | 2 |
| big data platform cost for myuhc | 2 |
| oracle cost for myuhc | 2 |

### Add a Webhook

This section provides information on how to enable and setup your own webhooks to your agent and intent. See the requirements below for your webhook in accepting a response from Rasa.

- Enable Webhooks by clicking on the gears icon of your Agent in the Agents Section

- Click on the switch for **Service Fullfilment** so it is now green as shown below
    - Enter your webhook's service url
    - If applicable, enter your webhook's basic auth credentials
- Navigate to your agent's intent that will be using a webhook and confirm **Enable webhook** switch is green as well.



OptumDev_Bot

Name

OptumDev_Bot

Client Secret

2190789d4078c3983f38df1623d77dfa

**Rasa Core Enablement** ○

**Service Fullfillment** ●

Service URL

Service Endpoint URL

Basic Auth Username

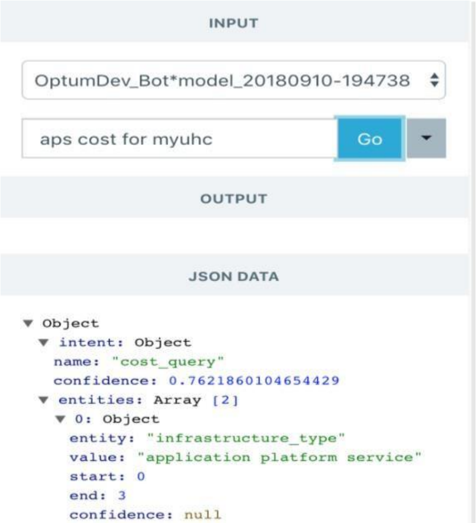
Basic Auth Username

Basic Auth Password

Basic Auth Password

**Test your Chatbot**

- Train your new chatbot
- Test your new chatbot model with the Rasa console UI Demo
- Enter "aps cost for myuhc" and observe the response and listed entity values
- Try other infrastructure types (ex. openshift, oracle, etc.) and UHG applications.
- Try new test expressions, with a similar intent. If the expression does not return the correct response, add more test data to your intent and retrain your chatbot.



INPUT

OptumDev_Bot*model_20180910-194738

aps cost for myuhc

OUTPUT

JSON DATA

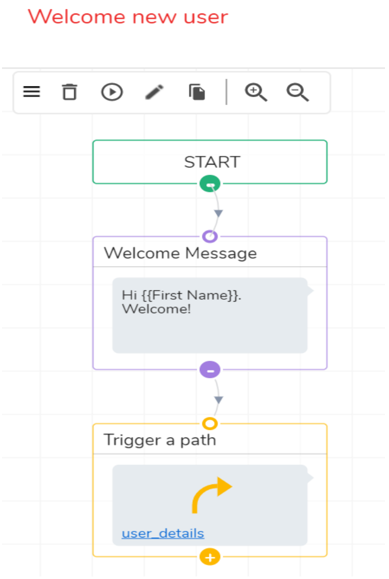
```
▼ Object
  ▼ intent: Object
    name: "cost_query"
    confidence: 0.7621860104654429
  ▼ entities: Array [2]
    ▼ 0: Object
      entity: "infrastructure_type"
      value: "application platform service"
      start: 0
      end: 3
      confidence: null
```

# HOW TO BUILD A CHATBOT WITH ENGATI.COM

To build a chatbot here, we need to develop paths or flows. These paths are analogous to intents. Each path symbolizes a specific set of operations to be performed by the bot. On engati platform, I developed a **bot which determines the weather conditions** in any city the user wants.
I used 5 paths for this purpose and named the bot as weather bot.

Details of the 5 paths are as follows:-

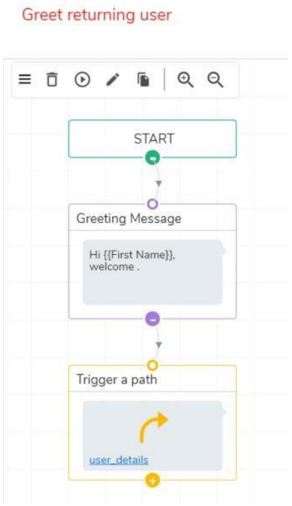
1. Welcome new user:



Hi {{First Name}}. Welcome!

This path, as the name suggests, is used to welcome a new user.
A welcome message can be written here which will be displayed in the beginning.
Here {{First_Name}} is an attribute .
Its value is taken from the user.
The path user_details which is triggered here is explained in the upcoming sections.

2. Greet returning user:



Greet returning user

START

Greeting Message

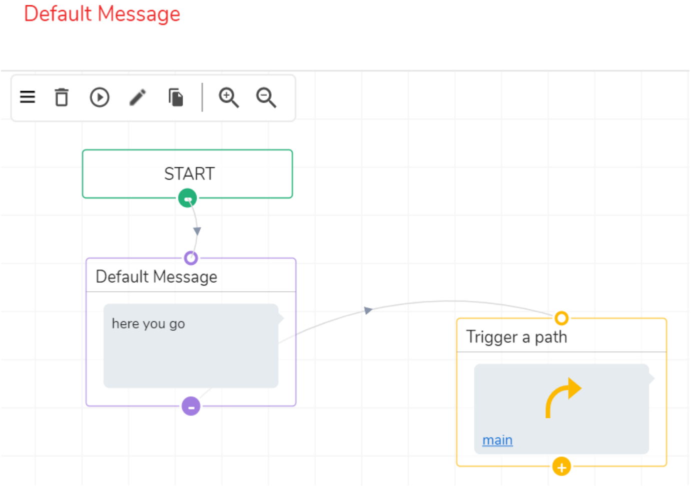
Hi {{First Name}}, welcome .

Trigger a path

user_details

Greet returning user path is used to greet a user who has returned to the conversation i.e. he is not a new user.
The path user_details which is triggered here is explained in the upcoming sections.

3. Default Message:



Default Message

START

Default Message

here you go

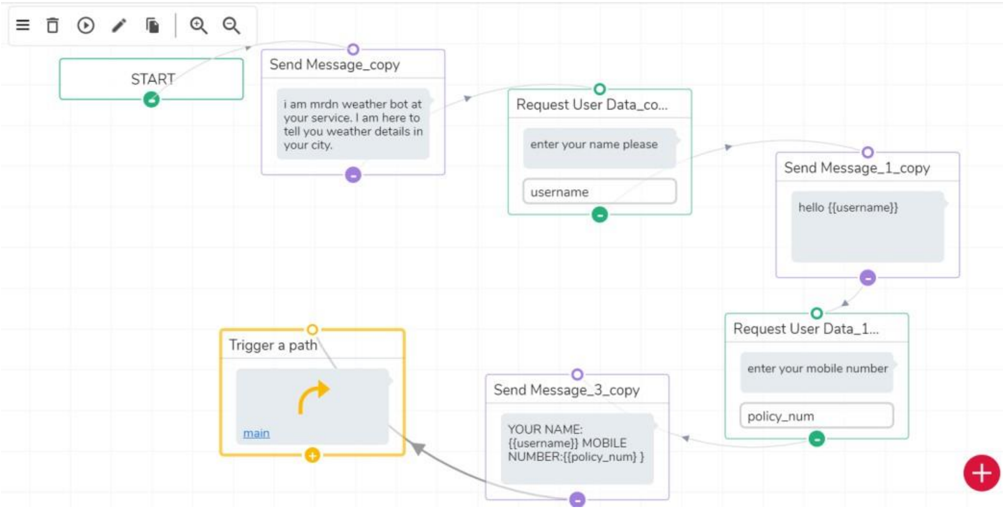
Trigger a path

main

This path will become active when user enters a value which the bot cannot understand or any invalid value.
For eg : If user enters "deli" for a city then Default Message path will be triggered.
I have linked the Default Message path to the main path which will be explained in coming sections.

4. User_details

This path is very important as it collects the user information (name and mobile number) and calls the main path.



i am mrdn weather bot at your service. I am here to tell you weather details in your city.

enter your name please

username

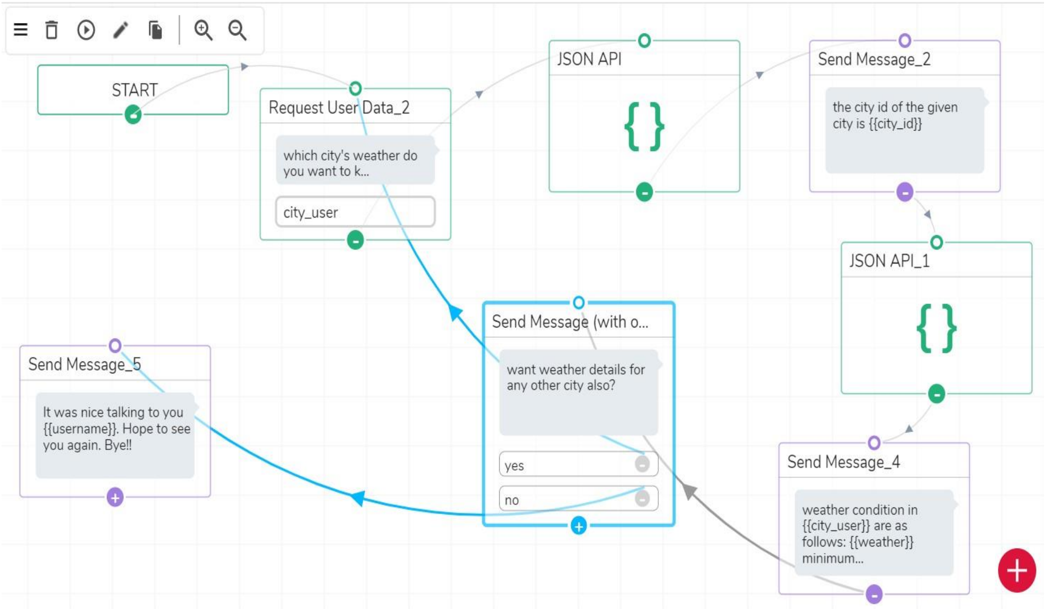
hello {{username}}

enter your mobile number

policy_num

YOUR NAME: {{username}} MOBILE NUMBER:{{policy_num} }

5. Main
   This is most important path of the entire logic. External API is integrated to the bot in this path. I have made 2 API calls here:
   a) https://www.metaweather.com/api/location/search/?query={{city_user}} (JSON API)
   b) https://www.metaweather.com/api/location/{{city_id}} (JSON API_1)



which city's weather do you want to k...

city_user

the city id of the given city is {{city_id}}

want weather details for any other city also?

yes

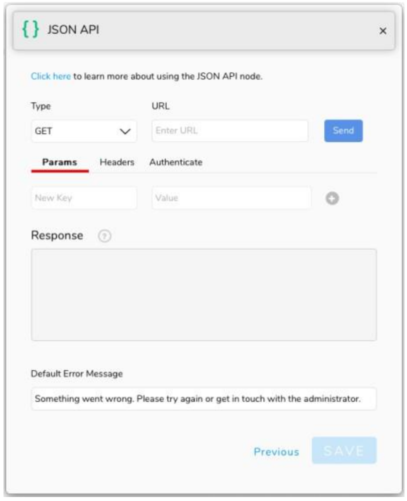
no

It was nice talking to you {{username}}. Hope to see you again. Bye!!

weather condition in {{city_user}} are as follows: {{weather}} minimum...

# Details of JSON API PLUGIN:

The JSON API Plugin provides flexibility for the bot admin to make HTTP requests, parse and store the responses into attributes and use the data across the platform. You can choose between GET, PUT and POST to make the HTTP requests.



Click here to learn more about using the JSON API node.

Type

GET

Default Error Message

Something went wrong. Please try again or get in touch with the administrator.

PARAMS

Click the Params tab to open the data editor for URL parameters. When you add key-value pairs, the JSON API Plugin automatically combines it with the URL while making the API call.

HEADERS

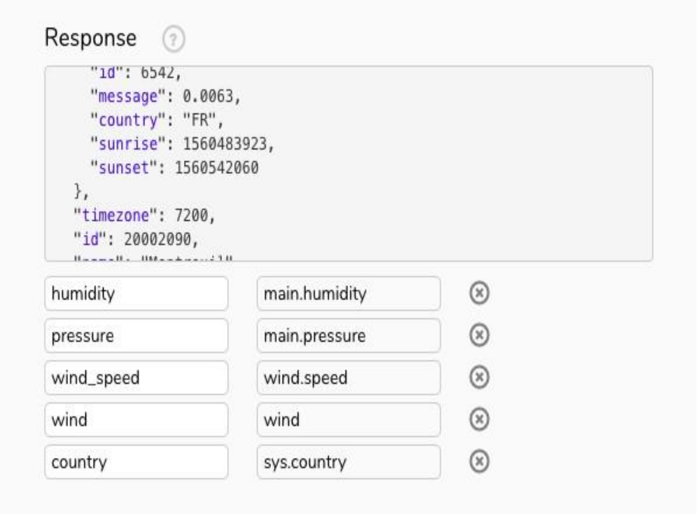
Clicking on the Headers tab shows the headers key-value editor. You can add your API request headers here for all method types.You can pass authentication tokens, app secret, passwords and other credentials as per the requirements of the end - point.

STORING API VALUES

To store values into attributes from responses from APIs:

- Click on the key from the response whose value you want to use in the flow
- The attribute is created right below the response box
- You can change the name of these attributes as per your desire

Thereafter, you can use these attributes anywhere in the flow.



```
    "id": 6542,
    "message": 0.0063,
    "country": "FR",
    "sunrise": 1560483923,
    "sunset": 1560542060
  },
  "timezone": 7200,
  "id": 20002090,
```

| | |
|---|---|
| humidity | main.humidity |
| pressure | main.pressure |
| wind_speed | wind.speed |
| wind | wind |
| country | sys.country |

# DEPLOYING THE BOT ON WEBSITE:

We can deploy our bot by adding this script in our HTML code, preferably before the closing body tag (</body>).

```
<script>!function(e,t,a)
{var c=e.head||e.getElementsByTagName("head")[0],n=e.createElement("script");n.async=!0,n.defer=!0,
n.type="text/javascript",n.src=t+"/static/js/chat_widget.js?config="+JSON.stringify(a),c.appendChild( n)}
(document,"https://app.engati.com",{bot_key:"d12cc3b743584c6e",bot_name:"Weather_Bot1",welco
me_msg:true,branding_key:"default"});</script>
```

# C++ CODE

Following is the C++ code for implementing basic functionalities of a conversational Chat Bot.

```
#include <iostream>
#include <string>
#include <vector>
#include <ctime>

const int MAX_RESP = 3;

typedef std::vector<std::string> vstring;

vstring find_match(std::string input);
void copy(const char* array[], vstring& v);


typedef struct {
        const char* input;
        const char* responses[MAX_RESP];
}record;
//char* x = "WHAT IS YOUR NAME";
record KnowledgeBase[] = {
        {"WHAT IS YOUR NAME",
        {"MY NAME IS CHATTERBOT2.",
         "YOU CAN CALL ME CHATTERBOT2.",
         "WHY DO YOU WANT TO KNOW MY NAME?"}
        },

        {"HI",
        {"HI THERE!",
         "HOW ARE YOU?",
         "HI!"}
        },

        {"HOW ARE YOU",
        {"I'M DOING FINE!",
        "I'M DOING WELL AND YOU?",
        "WHY DO YOU WANT TO KNOW HOW AM I DOING?"}
        },
        {"WHO ARE YOU",
        {"I'M AN A.I PROGRAM.",
         "I THINK THAT YOU KNOW WHO I'M.",
         "WHY ARE YOU ASKING?"}
        },

        {"ARE YOU INTELLIGENT",
        {"YES,OFCUORSE.",
         "WHAT DO YOU THINK?",
         "ACTUALY,I'M VERY INTELLIGENT!"}
        },

        {"ARE YOU REAL",
        {"DOES THAT QUESTION REALLY MATERS TO YOU?",
         "WHAT DO YOU MEAN BY THAT?",
         "I'M AS REAL AS I CAN BE."}
        }
};

size_t nKnowledgeBaseSize = sizeof(KnowledgeBase) /
sizeof(KnowledgeBase[0]);

int main() {
        srand((unsigned)time(NULL));

        std::string sInput = "";
        std::string sResponse = "";

        while (1) {
                std::cout << ">";
                std::getline(std::cin, sInput);
                vstring responses =  find_match(sInput);
                if (sInput == "BYE") {
                        std::cout << "IT WAS NICE TALKING TO YOU USER, SEE YOU
NEXTTIME!" << std::endl;
                        break;
                }
                else if (responses.size() == 0) {
                        std::cout << "I'M NOT SURE IF I  UNDERSTAND WHAT YOU
ARE TALKING ABOUT."
                                << std::endl;
                }
                else {
```

```
            int nSelection = rand() % MAX_RESP;
            sResponse = responses[nSelection]; std::cout <<
sResponse << std::endl;
        }
    }

    return 0;
}

// make a search for the user's input
// inside the database of the program
vstring find_match(std::string input) {
    vstring result;
    for (int i = 0; i < nKnowledgeBaseSize; ++i) {
        if (std::string(KnowledgeBase[i].input) == input) {
            copy(KnowledgeBase[i].responses, result);
            return result;
        }
    }
    return result;
}

void copy(const char* array[], vstring& v) {
    for (int i = 0; i < MAX_RESP; ++i) {
        v.push_back(array[i]);
    }

}
```

# FINAL TASK

## DESCRIPTION:

In the U.S healthcare provider (hospital) market, the revenue cycle management is a key business function that deals with the process of providers getting paid for the services they render to the patient.
The efficiency of the processes and technologies used in this area, play a vital role in maintaining the financial health of the provider organizations.
The project idea is one such improvement initiative that aims at enhancing the front office/patient access professionals to give cost estimates to the patients..
The idea is to use one of the latest technologies in the area of customer service management i.e. Chat Bots.
The proposed chatbot offers a convenient interface to the front office staff to instantly get real time cost estimates for the procedures, patient wants to take.
The approach is unique in itself as the chatbot uses an internal API that performs the estimation & calculation based on a wide range of factors and hence is a more accurate estimate.
Technologies/Platforms/Tools:
Microsoft Azure Cloud Computing Platform and Services
Visual Studio 2019
Engati Platform

## EXPECTED OUTCOME:

The final expected outcome is to develop the chatbot with a modernized user touchpoint and integrate it with the patient estimation as well as the other API's of the Meridian project, which internally connects with the back-end systems and provide the data as per the request.

After learning various techniques of building a conversational chat bot and integrating it with the REST API’s, I started building the bot for the Meridian project.

# Technologies/Platforms/Tools:

*Microsoft Azure Cloud Computing Platform and Services*

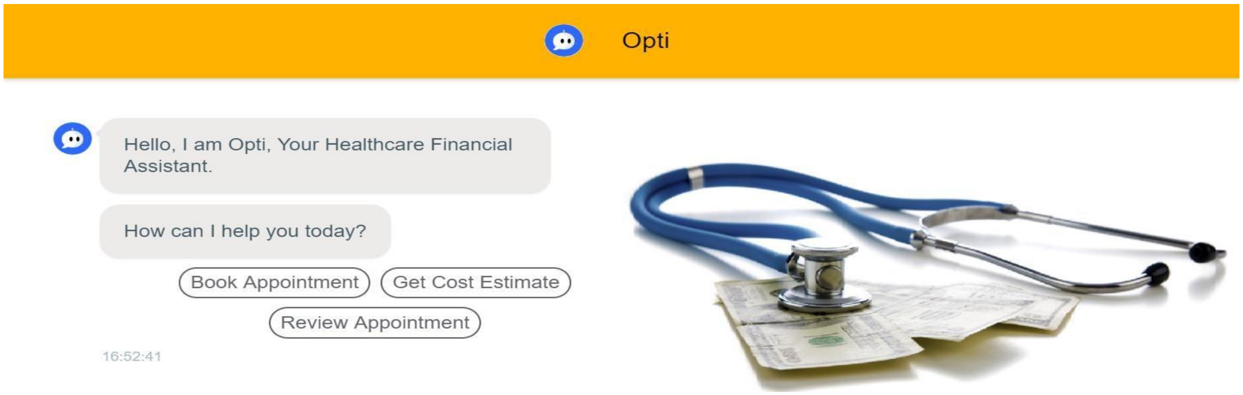
*Visual Studio 2019*

*Engati Platform*

# BOT NAME: OPTI

Various components of Opti are described below:-

## 1.) WELCOME NEW USER & GREET RETURNING USER :



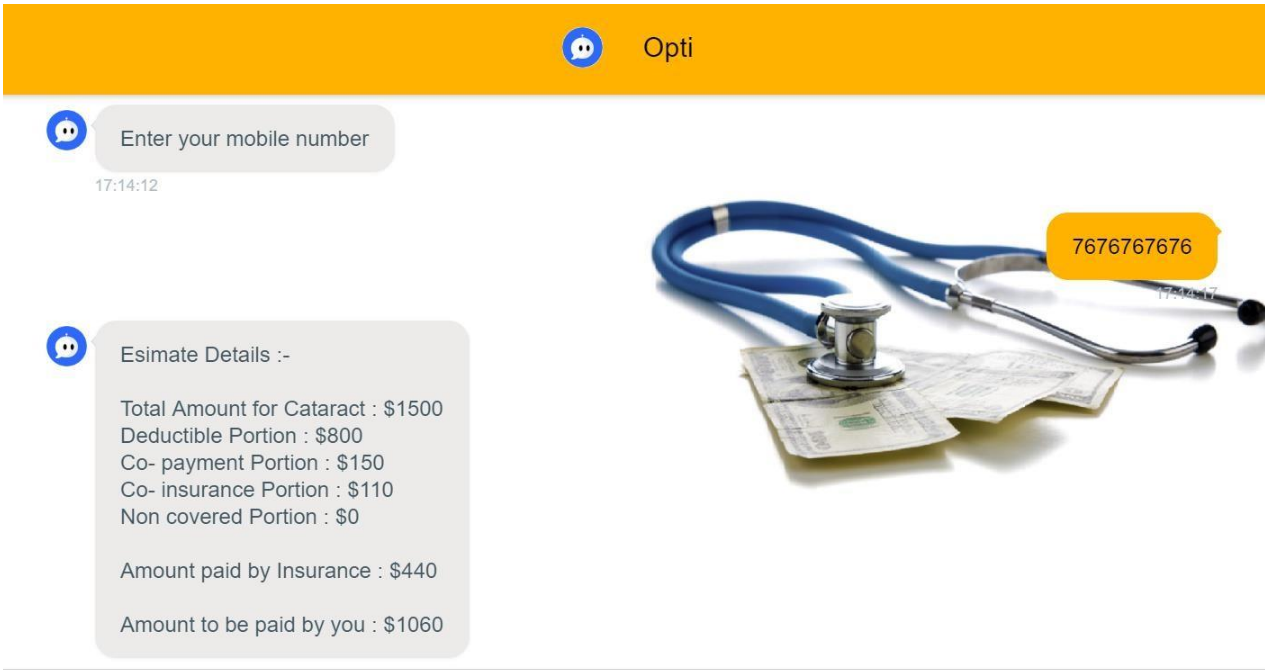
Opti

Hello, I am Opti, Your Healthcare Financial Assistant.

How can I help you today?

## 2.) REVIEWING PAST APPOINTMENTS :

### A.) ESIMATE DETAILS:



Opti

Enter your mobile number

17:14:12

7676767676

17:14:17

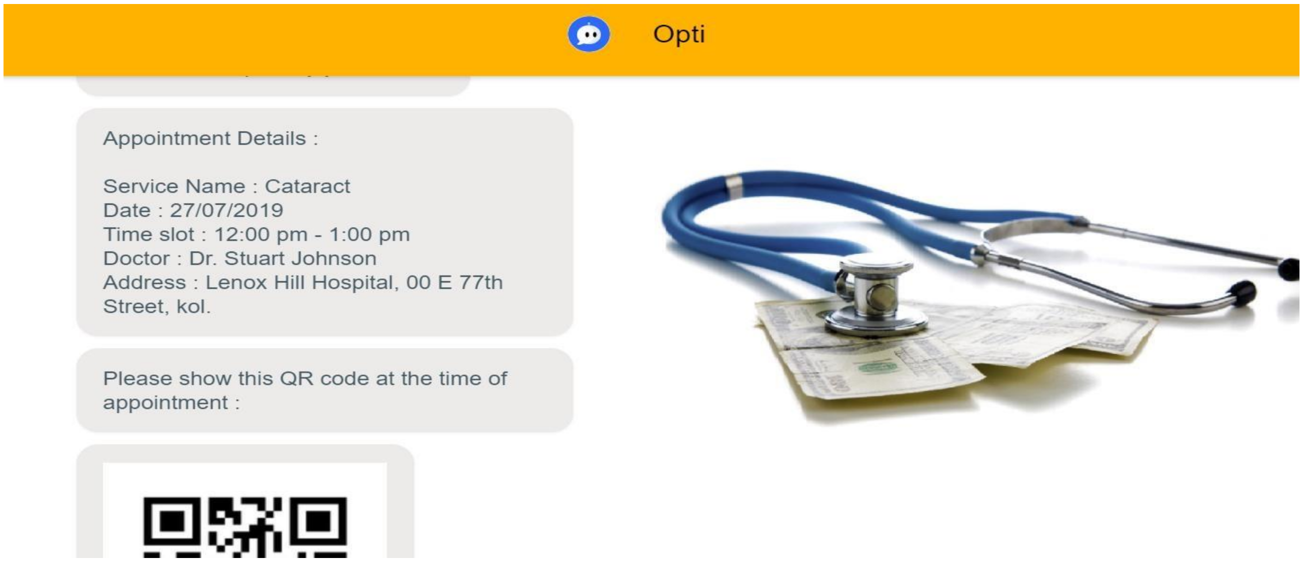
Esimate Details :-

Total Amount for Cataract : $1500
Deductible Portion : $800
Co- payment Portion : $150
Co- insurance Portion : $110
Non covered Portion : $0

Amount paid by Insurance : $440

Amount to be paid by you : $1060

B.) APPOINTMENT DETAILS :

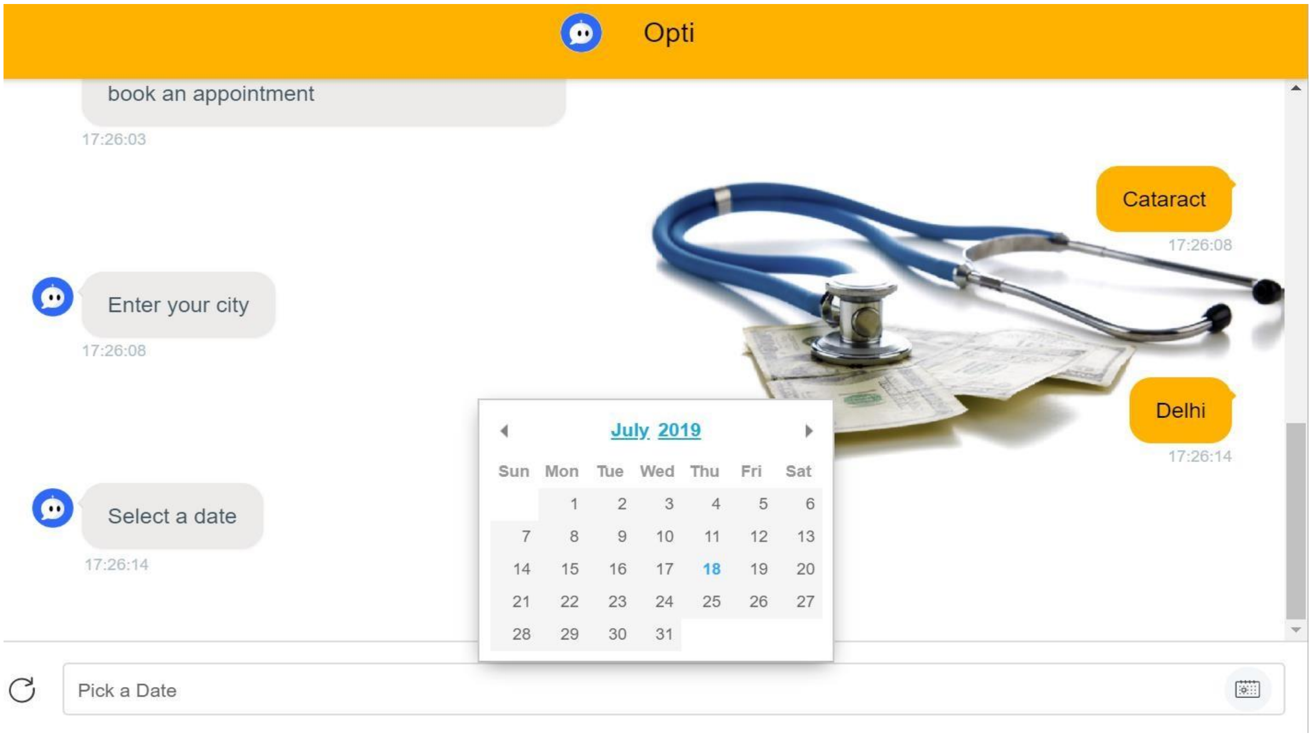


Appointment Details :

Service Name : Cataract
Date : 27/07/2019
Time slot : 12:00 pm - 1:00 pm
Doctor : Dr. Stuart Johnson
Address : Lenox Hill Hospital, 00 E 77th Street, kol.

Please show this QR code at the time of appointment :

3.) BOOKING AN APPOINTMENT :



book an appointment
17:26:03

Cataract
17:26:08

Enter your city
17:26:08

Delhi
17:26:14

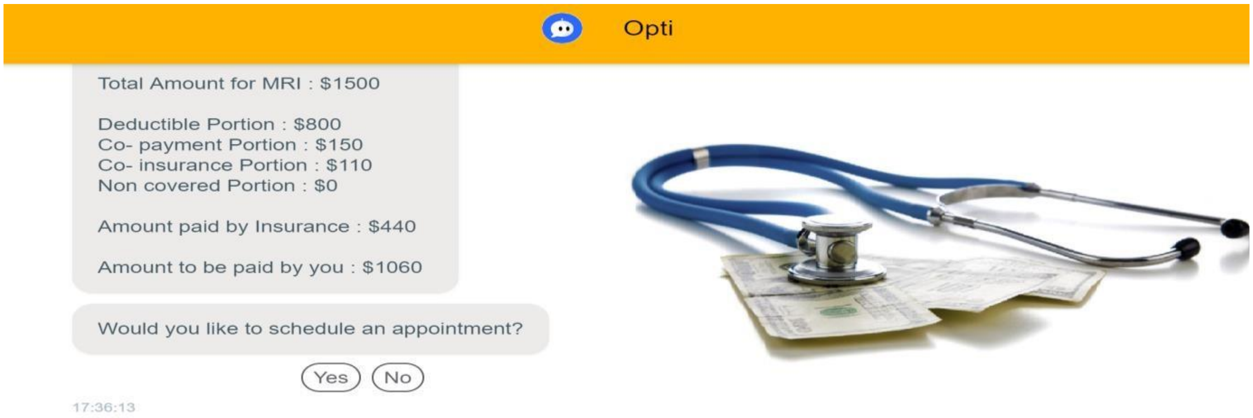
Select a date
17:26:14

July 2019

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 | | | |

Pick a Date

4.) GET COST ESTIMATE :



Total Amount for MRI : $1500

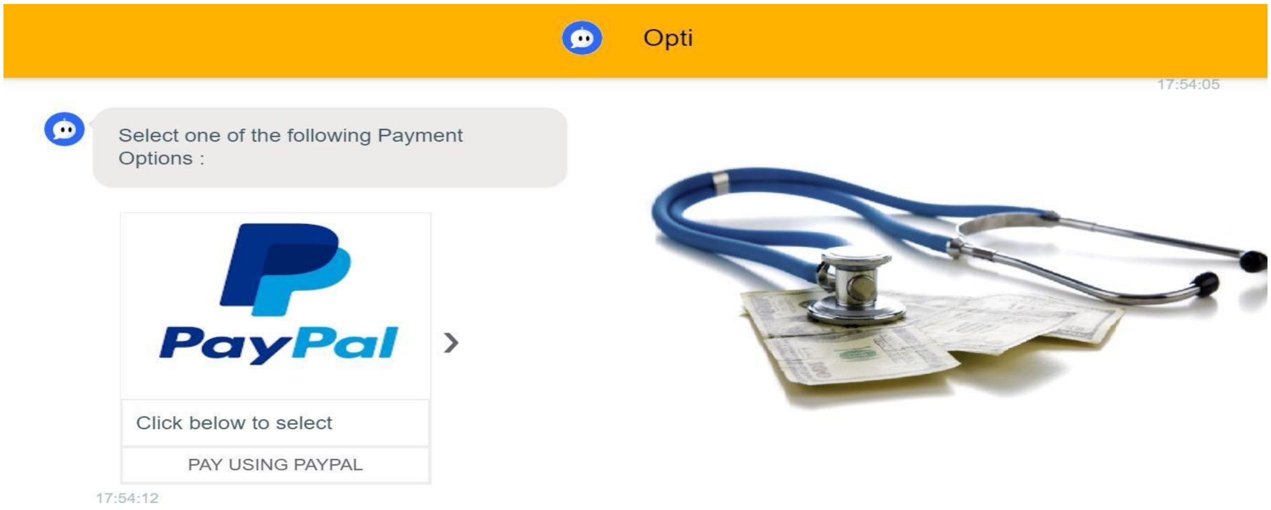
Deductible Portion : $800
Co- payment Portion : $150
Co- insurance Portion : $110
Non covered Portion : $0

Amount paid by Insurance : $440

Amount to be paid by you : $1060

Would you like to schedule an appointment?

5.) PAYMENT :



Select one of the following Payment Options :

6.) SAMPLE JSON INPUT :

```
{
  "inNetwork": true,
  "industryPayerId": "AUTO1",
"dateOfService": "2019-12-27",
"subscriber": {
  "dateOfBirth": "1978-08-13",
  "gender": "MALE",
  "memberId": "00918345566",
  "planEffectiveDate": "2018-02-01",
  "name": {
    "firstName": "JULIE",
    "lastNameOrOrgName": "EVANS",
    "middleName": "",
    "prefix": "",
    "suffix": ""
  }

},
"medicalServiceId": 1,
"placeOfService": 24,
"planId": "121212",
"provider": {
  "name": {
    "firstName": "",
    "lastNameOrOrgName": "HOUSTON MEDICAL IMAGING",
    "middleName": "",
    "prefix": "MD",
   "suffix": ""
  },
  "providerNpi": "1942385091"
 },
 "serviceLines": [
 {
                "cptCode": "91000",
                "modifier1": "M1",
        "modifier2": "M2",
                "modifier3": "",
                "modifier4": "",
                "billedCharge": 1500,
                "diagnosisCodes":["N99.9"]
        },
{
                "cptCode": "18064",
                "modifier1": "10",
                "modifier2": "11",
                "modifier3": "13",
     "modifier4": "15",
                "billedCharge": 2500,
                "diagnosisCodes":["LDC05"]
        }
 ],
  "response271":"ISA*00*          *00*          *ZZ*841162764      *30*621249087
*190205*0556*^*00501*036055635*0*P*:~GS*HB*841162764*621249087*20190205*0556*1*X*005010
X279A1~ST*271*000000001*005010X279A1~BHT*0022*11*295337315*20190205*0656~HL*1**20*1~N
M1*PR*2*GEHA*****PI*440545275~HL*2*1*21*1~NM1*1P*1*TRAUTMANN*PAUL****XX*1942385091~
HL*3*2*22*0~NM1*IL*1*Sosa*Roxanne*M***MI*23751754~REF*HJ*0~REF*6P*10000001*Federal
Employee Health Benefit Program~N3*354 Rio Azul~N4*Pipe
Creek*TX*780636175~DMG*D8*19780813*F~EB*1*FAM*30*PR*Standard Option Medical
FEHB~DTP*291*D8*20150622~EB*P***********W~MSG*UNLESS OTHERWISE REQUIRED BY LAW,
THIS NOTICE IS NOT A GUARANTEE OF PAYMENT. BENEFITS ARE SUBJECT TO ALL CONTRACT
LIMITS AND THE MEMBER'S STATUS ON THE DATE OF SERVICE. ACCUMULATED AMOUNTS
SUCH AS DEDUCTIBLE MAY CHANGE AS ADDITIONAL CLAIMS ARE
PROCESSED.~EB*C*IND*30***23*350*****W~EB*C*IND*30***29*341.1*****W~III*ZZ*12~EB*C*FAM*30
***23*700*****W~EB*C*FAM*30***29*219.17*****W~III*ZZ*10~EB*C*FAM*30***23*700*****W~EB*C*FA
M*30***29*219.17*****W~III*ZZ*11~EB*C*FAM*30***23*700*****W~EB*C*FAM*30***29*219.17*****W~II
I*ZZ*24~EB*G*FAM*30***23*6000*****Y~EB*G*FAM*30***29*5449.96*****Y~III*ZZ*12~EB*G*FAM*30***
23*6000*****Y~EB*G*FAM*30***29*5449.96*****Y~III*ZZ*11~EB*G*FAM*30***23*6000*****Y~EB*G*FA
M*30***29*5449.96*****Y~III*ZZ*10~EB*G*FAM*30***23*6000*****Y~EB*G*FAM*30***29*5449.96*****Y
~III*ZZ*24~EB*A*IND*5*****.22****Y~III*ZZ*22~EB*A*IND*5*****.21****Y~III*ZZ*21~EB*A*IND*5*****.20*
***Y~III*ZZ*20~EB*A*IND*30*****.12****Y~III*ZZ*12~EB*A*IND*30*****.11****Y~III*ZZ*11~EB*A*IND*30*
****.10****Y~III*ZZ*10~EB*A*IND*98*****.32****Y~III*ZZ*32~EB*A*IND*98*****.31****Y~III*ZZ*31~EB*A*I
ND*98*****.30****Y~III*ZZ*30~SE*42*000000001~GE*1*1~IEA*1*036055635~",

  "taxonomyCode": "1212121212" }
```

7.) <u>SAMPLE JSON OUTPUT :</u>

```
{
  "transactionUuid": "a172a6a2-9262-4dea-8d47-165eb4c916b5",
  "benefits": {
   "outOfPocket": {
    "remaining": 5449.96,
    "family": {
     "max": 6000,
     "met": 550.04,
     "remaining": 5449.96
    },
    "individual": {
     "max": null,
     "met": null,
     "remaining": null
    }
   },
   "deductible": {
    "remaining": 219.17,
    "family": {
      "max": 700,
      "met": 480.83,
      "remaining": 219.17
    },
   "individual": {
      "max": 350,
      "met": 8.9,
      "remaining": 341.1
    }
  },
  "copayment": 0,
  "coinsuranceRate": 0.11,
  "limitationAmount": 0
},
"estimate": {
  "patientResponsibility": {
  "deductiblePortion": 219.17,
    "copaymentPortion": 0,
    "coinsurancePortion": 415.8913,
    "nonCoveredPortion": 0,
    "amount": 635.0613
  },
  "payerResponsibility": {
    "amount": 3364.9387
  },
  "amount": 4000
},
"serviceLines": [
  {
    "cptCode": "91000",
    "modifier1": "M1",
    "modifier2": "M2",
    "modifier3": "",
    "modifier4": "",
      "diagnosisCodes": [
        {
          "value": "N99.9"
        }
      ],
      "patientResponsibility": 360.0613,
      "payerResponsibility": 1139.9387,
     "wellnessMatched": false,
     "priceType": "BILLED_CHARGE",
     "aceEditExists": false
    },
    {
      "cptCode": "18064",
      "modifier1":"10",
      "modifier2":"11",
      "modifier3":"13",
      "modifier4": "15",
      "diagnosisCodes": [
        {
          "value": "LDC05"
        }
      ],
      "patientResponsibility": 275,
      "payerResponsibility": 2225,
      "wellnessMatched": false,
      "priceType": "BILLED_CHARGE",
      "aceEditExists": false
    }
  ],
  "estimateType": "SIMPLE_ESTIMATE"
}
```

I developed the bot and tested it with some basics API's of Meridian project and it gave accurate results.

The final step was to make the bot capable to work with the 'SIMPLE ESTIMATE API' which takes the information from the user like name, insurance policy number, DOB, medical service required and a few more and produce the cost estimate to be paid by the user for that particular medical service.

The bot was to retrieve data from the internal back-end databases and provide correct results. Also, it was expected that the bot should be capable enough to adapt to the changes in the datasets, all by itself.

Finally, I was able to design a chat bot with all the necessary requirements and it worked absolutely fine on integration with the 'SIMPLE ESTIMATE API' and other API's as well.

The project got completed on 16th July,2019 with the final demo in front of the entire Meridian-India team and after that I worked on some minor modifications of the the chat bot regarding visual effects and additional features like small talk, enabling user to change his previous response, auto pop-up of the chat bot widget on the HTML page with a pop-up message etc.

**The REST API's mentioned above are confidential and cannot be shared with anyone. Any further details regarding the API's and the project will be considered as a violation of the rules of OPTUM GLOBAL SOLUTIONS.**

# REFERENCES

- https://www.expertsystem.com/chatbot/
- https://www.infoworld.com/article/3244546/9-chatbot-building-tools.html
- https://chatbotslife.com/dialogflow-restaurant-bot-tutorial-1-45ce1d3c0ab5
- https://www.analyticsvidhya.com/blog/2019/04/learn-build-chatbot-rasa-nlp-ipl/
- https://engati.com/
- https://knowledgebase.engati.com/kb/
- https://visualstudio.microsoft.com/
- https://www.google.com/search?q=azure+portal&oq=azure&aqs=chrome.1.69i57j0l3j69i61l2.2587j0j4&sourceid=chrome&ie=UTF-8
- https://swagger.io/