PROJECT REPORT

Intelligent Customer Help Desk with Smart Document Understanding

Category: Artificial Intelligence

Application Id: SPS_APL_20200000678

Project Id: SPS_PRO_99

Internship at SmartInternz

Pooja Milind Tambe

Pooja123tambe@gmail.com

1.INTRODUCTION

1.1 Overview:

The typical customer care chatbot can answer simple questions, such as store locations and hours, directions, and maybe even making appointments. When a question falls outside of the scope of the pre-determined question set, the option is typically to tell the customer the question isn't valid or offer to speak to a real person.

To Overcome this problem, the chatbot is made intelligent by using Artificial Intelligence. In this project Watson Discovery service is used to learn a document such as user manual, a book,other reference pdfs etc.

1.2 Purpose:

In this project, there will be another option. If the customer question is about the operation of a device, the application shall pass the question onto Watson Discovery Service, which has been pre-loaded with the device's owners manual. Hence system can return relevant sections of the owners manual to help solve our customers' problems related to that application rather than showing invalid answer.

The project has used the Smart Document Understanding feature of Watson Discovery to train it on what text in the owners manual is important and what is not. This will improve the answers returned from the queries.

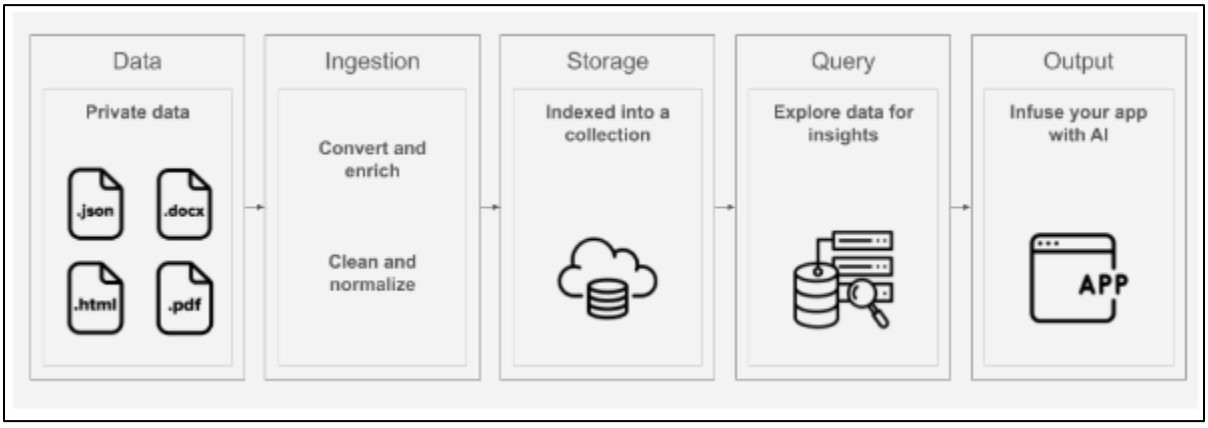
Here, we have used product heater. User who wants to know about basic information and working of heater asks query to chatbot.

2.LITERATURE SURVEY

2.1 Watson Discovery:

Using IBM Watson Discovery, one can ingest, normalize, enrich, and search your unstructured data (JSON, HTML, PDF, Word, and more) with speed and accuracy. It packages core Watson APIs such as Natural Language Understanding and Document Conversion along with UI tools that enable you to easily upload, enrich, and index large collections of private or public data.

The following image shows a high-level view of all of the components that make up the Discovery pipeline.



Discovery has a powerful analytics engine that provides cognitive enrichments and insights into your data. With built-in natural language processing (NLP) capabilities, it can extract enrichments from a wide range of document types.

All of the data content is stored and enriched within a Discovery collection. The data does not require any specific structure and can come from multiple public and private data sources. Every Discovery environment comes with a pre-enriched data collection named Watson Discovery News.

Steps to follow:

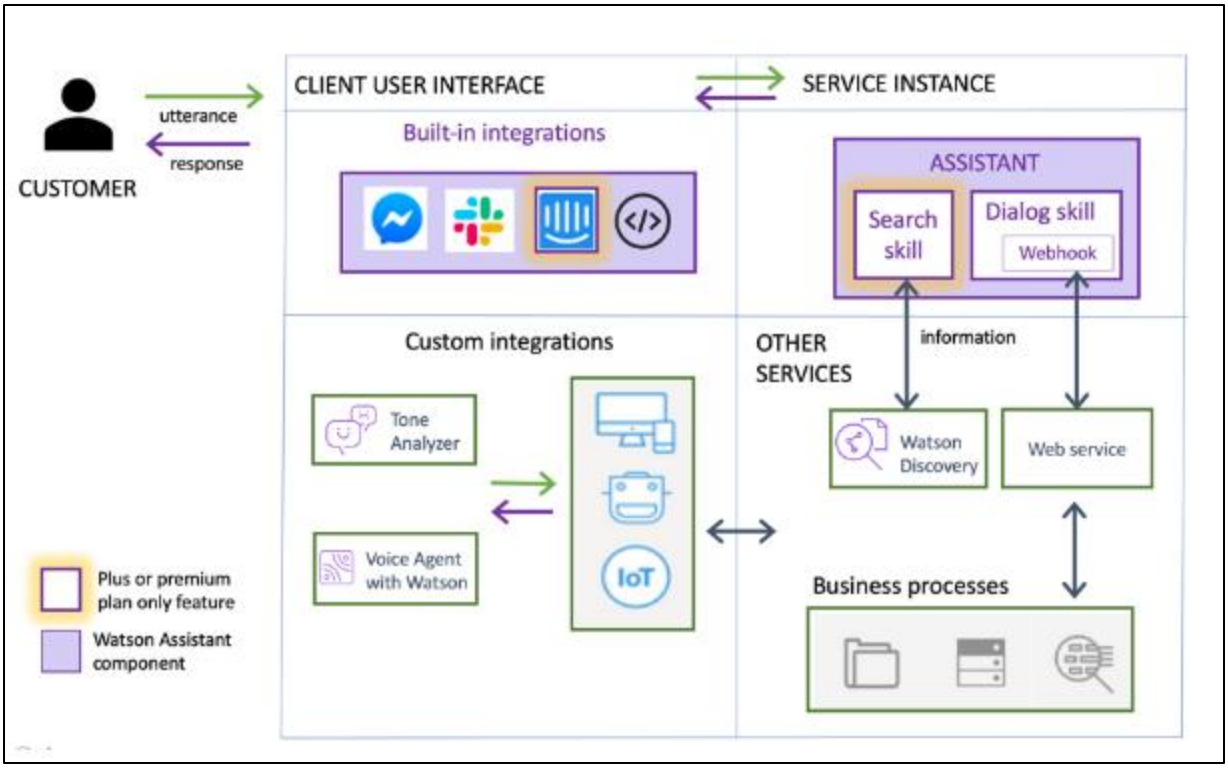
- *The need to search thousands of product reviews at once*: Create a Discovery collection and build a UI to query the collection and graph the sentiment over time.
- *The need to programmatically find text within a document*: Use the passage retrieval feature of Discovery to create an FAQ chatbot.
- *There are thousands of documents in different formats and you need to organize them logically*: Use Discovery to pull out keywords, concepts, and relationships to sort them.

2.2 Watson Assistant:

Watson Assistant is a conversation AI platform that helps you provide customers fast, straightforward and accurate answers to their questions, across any application, device or channel. By addressing common customer inquiries, Watson Assistant reduces the cost of customer interactions, helping your agents focus on complex use cases – not repetitive responses.

With IBM Watson Assistant, you can build conversational interfaces into any application, device, or channel. Most virtual assistants try to mimic human interactions, but Watson Assistant is more. Watson Assistant knows when to search for an answer from a knowledge base, when to ask for clarity, and when to direct you to a human.

- Users interact with the assistant through one or more of these integration points:
- A virtual assistant that you publish directly to an existing social media messaging platform, such as Slack or Facebook Messenger.
- A custom application that you develop, such as a mobile app or a robot with a voice interface.
- The assistant receives user input and routes it to the dialog skill.
- The dialog skill interprets the user input further, then directs the flow of the conversation. The dialog gathers any information it needs to respond or perform a transaction on the user's behalf.
- Any questions that cannot be answered by the dialog skill are sent to the search skill, which finds relevant answers by searching the company knowledge bases that you configure for the purpose.



2.3 IBM cloud function:

IBM Cloud Functions is a distributed compute service that executes application logic in response to requests from web or mobile apps. You can set up specific actions to occur based on HTTP-based API requests from web apps or mobile apps, and from event-based requests from services like Cloudant.

With IBM Cloud™ Functions, you can create stateless code snippets, called actions, that are set to perform one specific task.

2.4Node-Red Application:

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click. Node-RED is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal

to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

3.THEORITICAL ANALYSIS

3.1 Steps to make chatbot:

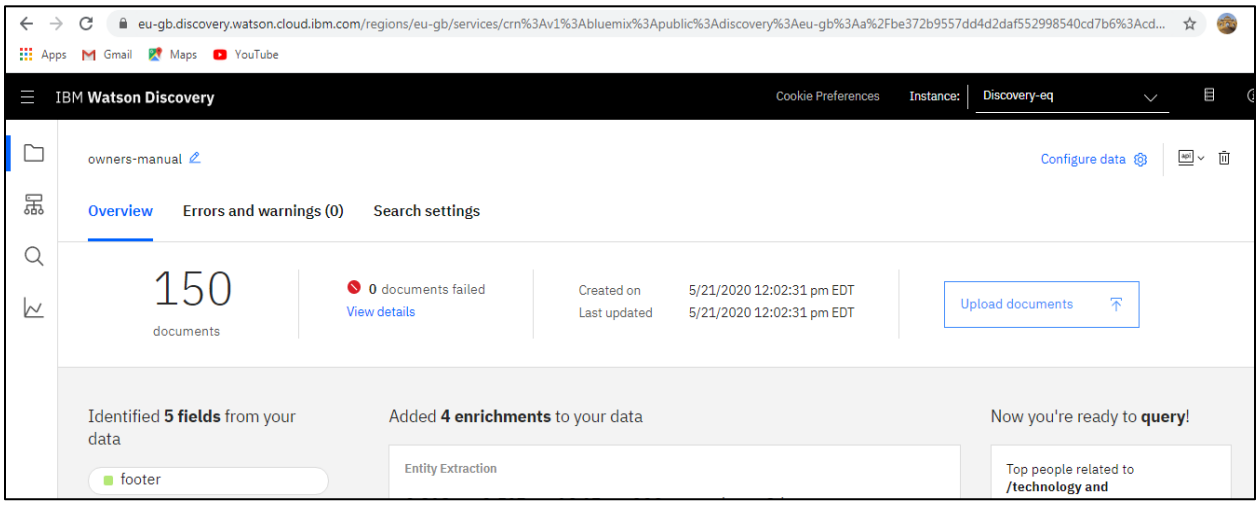
1.**Create IBM cloud services:**

Watson Discovery

Watson Assistant

2.Configure Watson Discovery: Import the document

Launch the Watson Discovery tool and create a new data collection by selecting the Upload your own data option. Give the data collection a unique name. Select and upload the ecobee3_UserGuide.pdf file located in the your device.
Now let's apply SDU to our document to see if we can generate some better query responses.



3. Create IBM Cloud Functions action

Now let's create the web action that will make queries against our Discovery collection.
Select the Functions card from catalog of IBM cloud.
From the Functions main panel, click on the Actions tab. Then click on Create.
Once your action is created, select the Parameters tab, add the following keys:

- url
- environment_id
- collection_id
- iam_apikey

For values, please use the values associated with the Discovery service you created in the previous step.

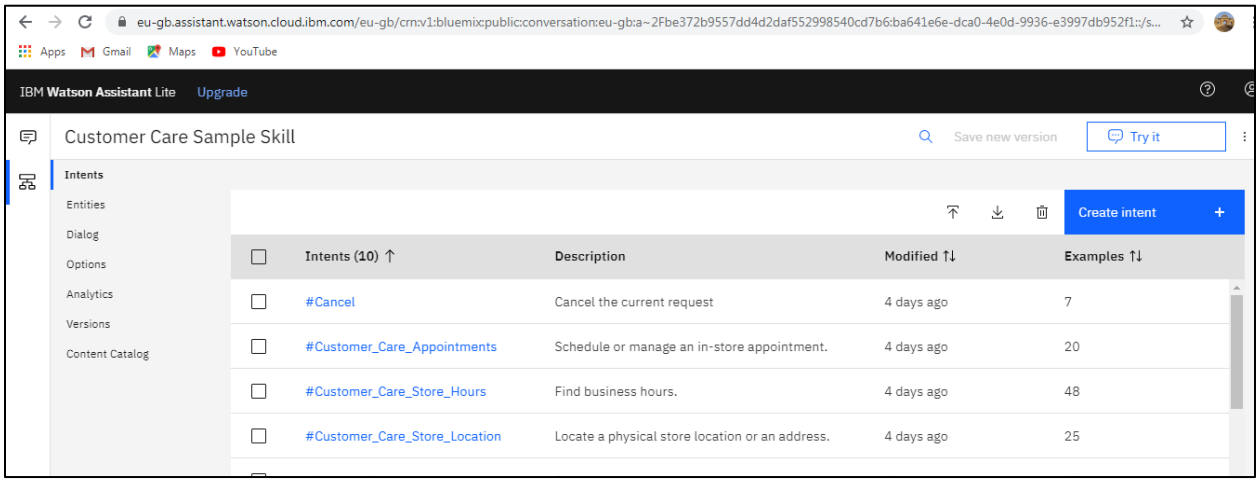
Now that the credentials are set, return to the Code panel and press the Invoke button again. Now you should see actual results returned from the Discovery service.

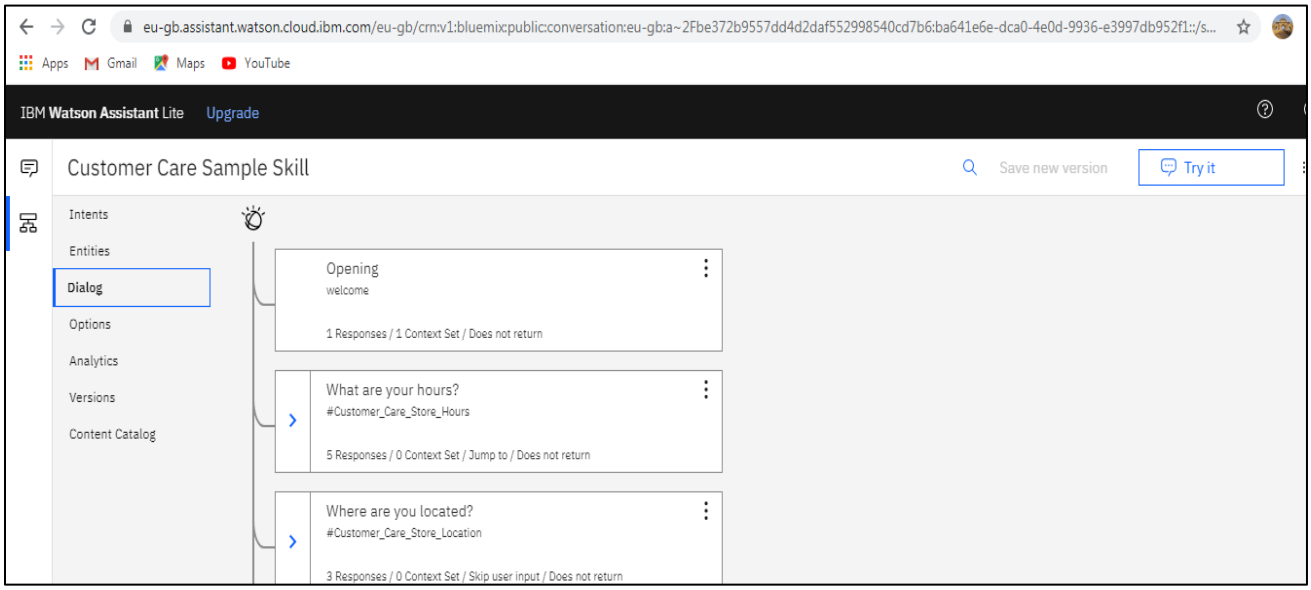
4. Configure Watson Assistant

Launch the Watson Assistant tool and create a new dialog skill. Select the Use sample skill option as your starting point.
This dialog skill contains all of the nodes needed to have a typical call center conversation with a user.

Add new intent

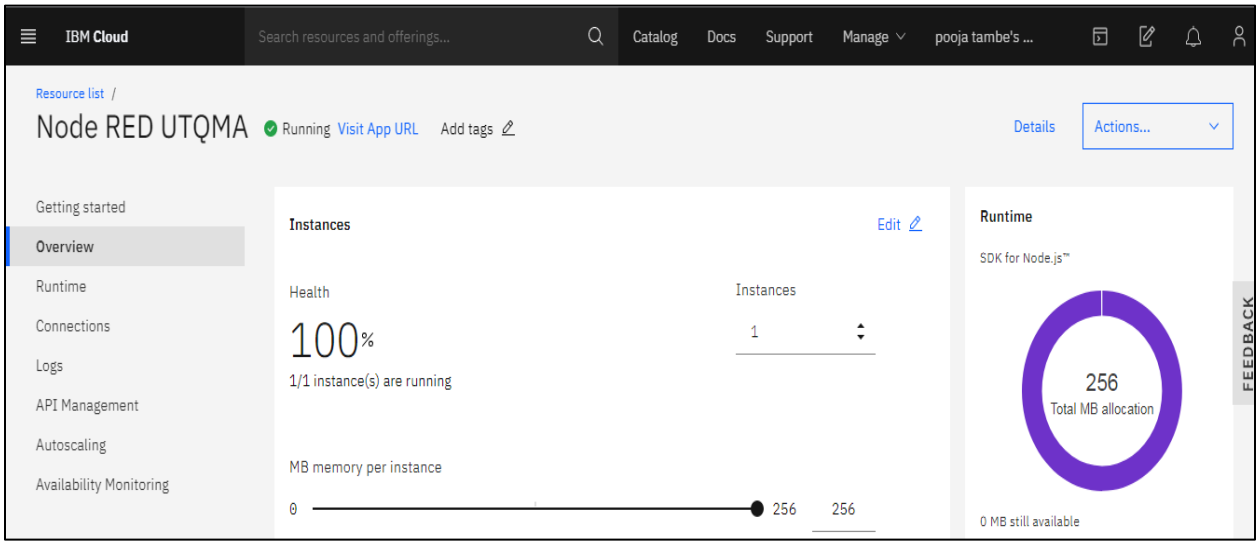
Create a new intent that can detect when the user is asking about operating the Ecobee thermostat.
From the Customer Care Sample Skill panel, select the Intents tab.
Click the Create intent button.
Name the intent #Product_Information, and enter the example questions to be associated with it. Create new dialog node. Test in Assistant Tooling query from given example question.

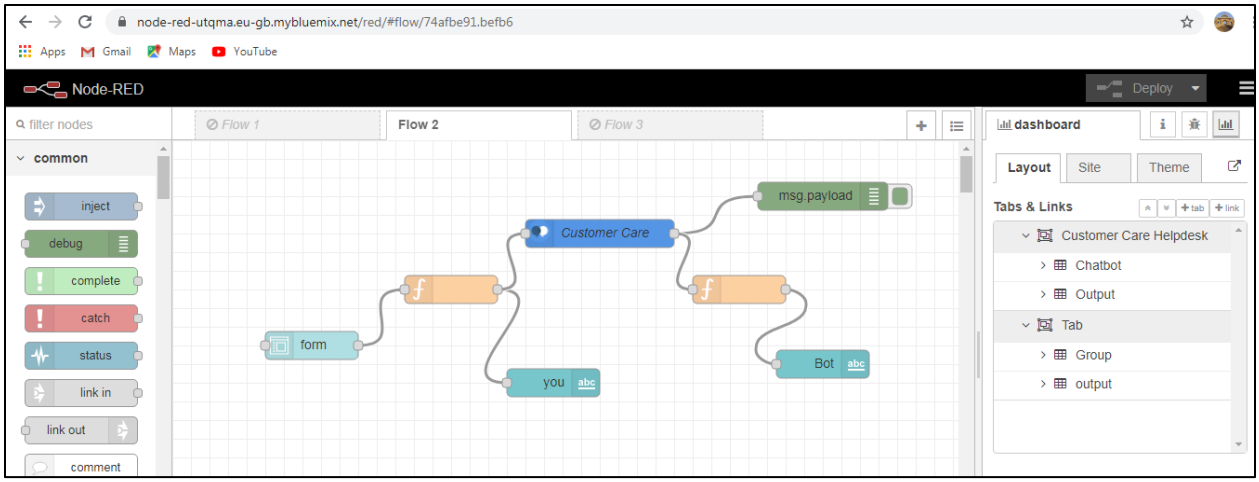




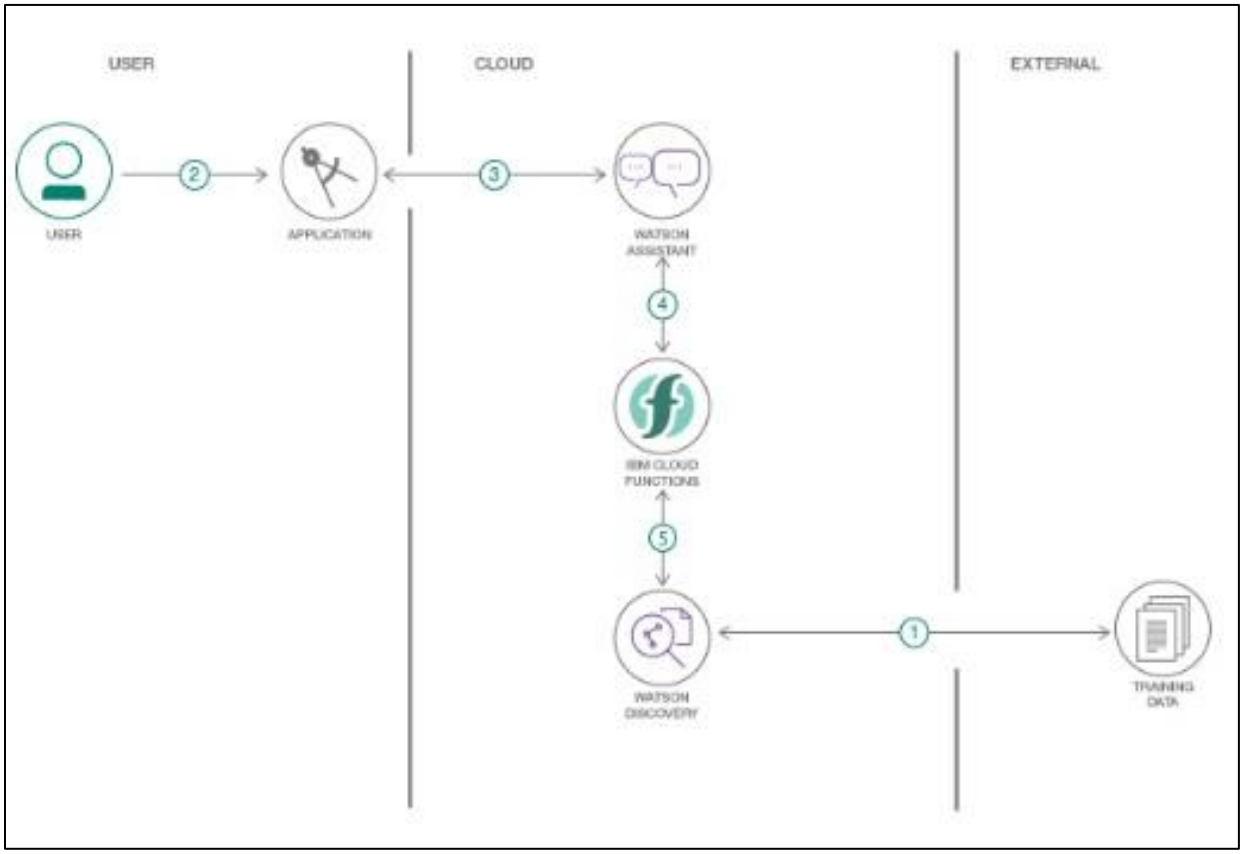
5. Node-Red Application

From IBM cloud catalog select Node-red starter application kit. Create Node- red application . Now create flow of application on Node-red platform using various nodes such as function, form, assistant, debug etc.
Configure each node according to the given application and then deploy it.



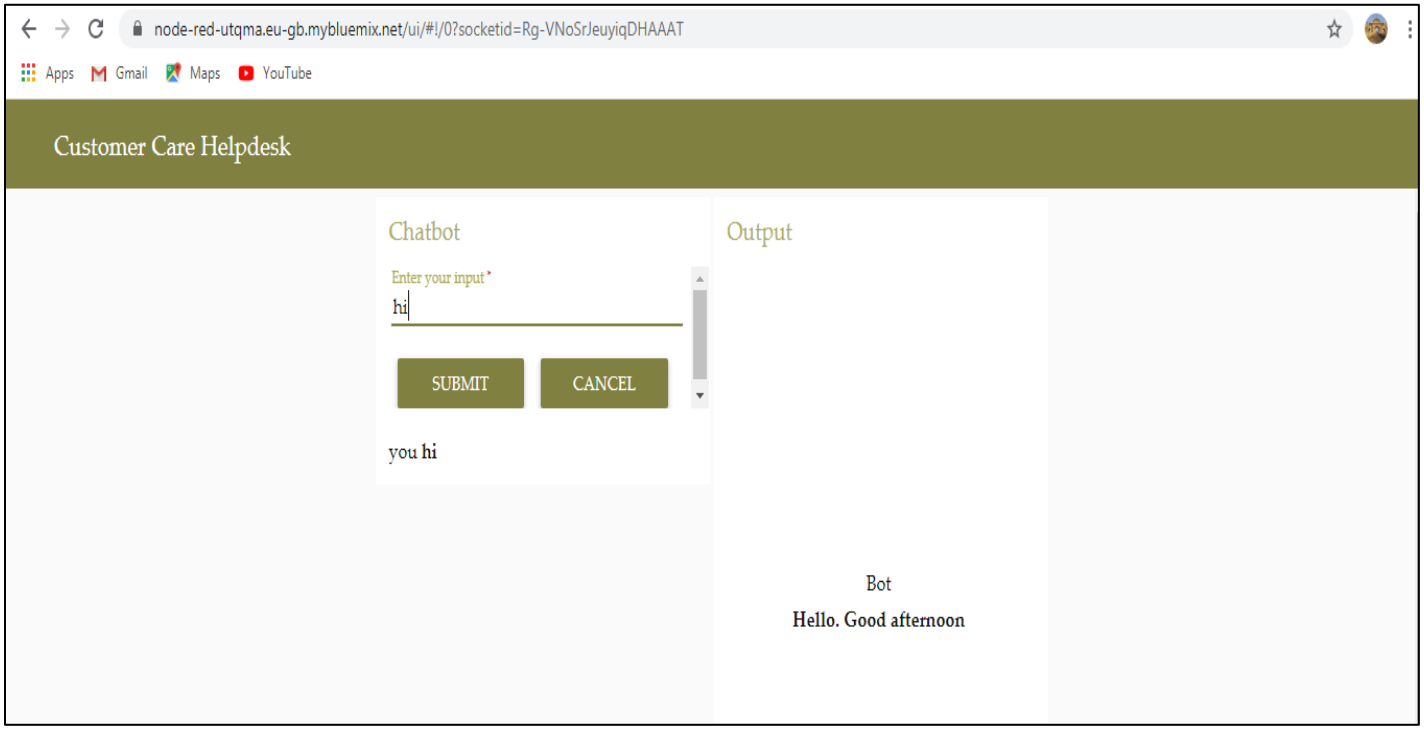

node-red-utqma.eu-gb.mybluemix.net/red/#flow/74afbe91.befb6
Apps
Gmail
Maps
YouTube
Node-RED
Deploy
filter nodes
Flow 1
Flow 2
Flow 3
common
inject
debug
complete
catch
status
link in
link out
comment
msg.payload
Customer Care
form
you
Bot
dashboard
Layout
Site
Theme
Tabs & Links
Customer Care Helpdesk
Chatbot
Output
Tab
Group
output

4.FLOW DIAGRAM



5.RESULT

- After deploying on Node-red window, the following window will be shown which is final outcome desired.
- Here, we can test its working by entering required input, then press submit.
- The chatbot answers query asked.



6.CONCLUSION

The project makes Intelligent and smart system using Artificial Intelligence, Machine learning approaches. Understanding document feature given by Watson Discovery learns provided document and performs inferencing based on asked query by customer.

The AI platform used is Watson Assistant, which responds to customer based on output from Watson Discovery. It contains various skills by which we created dialog skills.

Node-Red application ties up everything using flow codes and creates web interface.

APPENDIX

A. Code

This code is used while creating function in IBM cloud.

```
/**
 *
 * @param {object} params
 * @param {string} params.iam_apikey
 * @param {string} params.url
 * @param {string} params.username
 * @param {string} params.password
 * @param {string} params.environment_id
 * @param {string} params.collection_id
 * @param {string} params.configuration_id
 * @param {string} params.input
 *
 * @return {object}
 *
 */

const assert = require('assert');
const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');

/**
 *
 * main() will be run when you invoke this action
 *
 * @param Cloud Functions actions accept a single parameter, which must be a JSON object.
 *
 * @return The output of this action, which must be a JSON object.
 *
 */
function main(params) {
```

```
return new Promise(function (resolve, reject) {

  let discovery;

  if (params.iam_apikey){
    discovery = new DiscoveryV1({
      'iam_apikey': params.iam_apikey,
      'url': params.url,
      'version': '2019-03-25'
    });
  }
  else {
    discovery = new DiscoveryV1({
      'username': params.username,
      'password': params.password,
      'url': params.url,
      'version': '2019-03-25'
    });
  }

  discovery.query({
    'environment_id': params.environment_id,
    'collection_id': params.collection_id,
    'natural_language_query': params.input,
    'passages': true,
    'count': 3,
    'passages_count': 3
  }, function(err, data) {
    if (err) {
      return reject(err);
    }
    return resolve(data);
  });
 });
}
```