1. **INTRODUCTION**
   1. **Overview**

Human Interaction with machines has increased since Alan Turing created the first computer. Chat bots and voice assistants are 2 of the key reasons of the growth. This project is focusing on creating a web-based chatbot to be used as an initial customer care for a product called ecobee3. With automation looking to takeover manual labour and factory type jobs, chatbots are starting to make their way into the customer service sector. Besides customer care the chatbot is well equipped to have friendly conversations with its users. The chatbot is built on IBM cloud as an internship project given by Smart Bridge. It uses Node Red as backend.

* 1. **Purpose**

The purpose of this project is to showcase the power of chatbots and how there can be an alternative to using this application on a website. The chatbots should be easy to use, respond in a timely fashion and be all round user friendly. The bots should make the users interaction as easy and fast as possible to ensure that the users time is not wasted and that they get what they want without any difficulty or misunderstanding from the bot. The conversation should flow and always keep the user in control of the conversation. Users should come away from their experience with the chatbot and think that it was a fun, easy to use and straightforward interaction that would encourage them to come back without any hesitation.

1. **LITERATURE SURVEY**
   1. **Existing problem**

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 predetermined question set, the option is typically to tell the customer the question isn’t valid or offer to speak to a real person.

* 1. **Proposed solution**

IBM provides a resource called Watson Discovery. With IBM Watson Discovery, we can ingest, normalize, enrich, and search our 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. If the chatbot is asked a query which is outside its scope the query is processed by the Watson Discovery API.

1. **THEORITICAL ANALYSIS**
   1. **Block diagram**

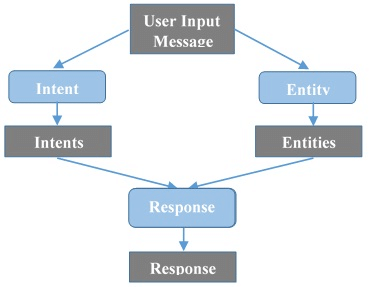
****

Fig. 1 Chatbot Working

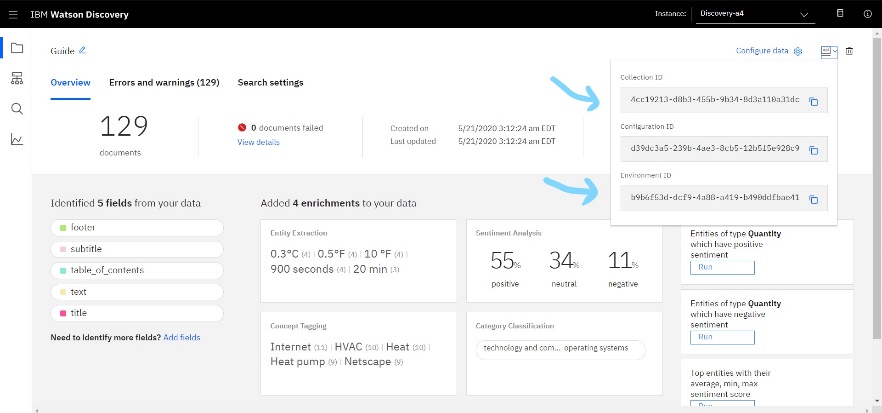
* 1. **Hardware / Software designing**

1. Creating IBM Cloud services:

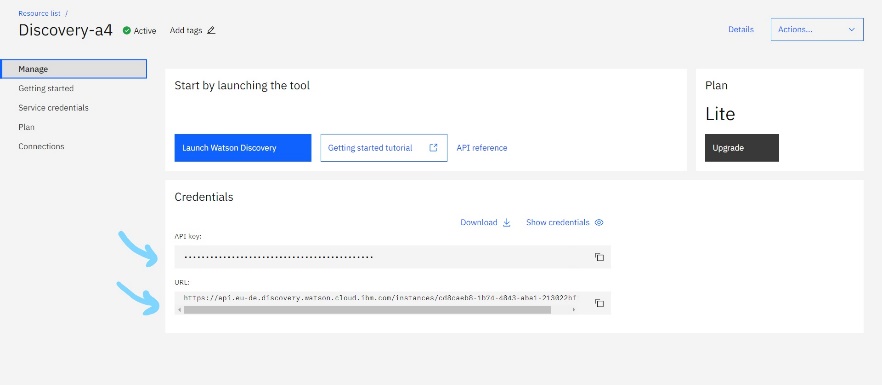
* Watson Discovery
* Watson Assistant
* Node Red

1. Configure Watson Discovery:

* 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. When prompted, select and upload ecobee3\_UserGuide.pdf (or any other document as required) file located in the data directory of your local environment.
* To Annotate with Smart Document Understanding (SDU) on the document click the Configure data button (located in the top right corner) to start the SDU process. The goal is to annotate all of the pages in the document so Discovery can learn what text is important, and what text can be ignored.
* As the pages are annotated one at a time, Discovery is learning and automatically updating the upcoming pages. More pages annotated, better the model will be trained.
* A few credentials of the trained Discovery model will be in use to access Discovery collection. The credentials are Collection ID, Environment ID, URL, and API key. The Collection ID and Environment ID values can be found by clicking the drop-down button located at the top right side of the collection panel.

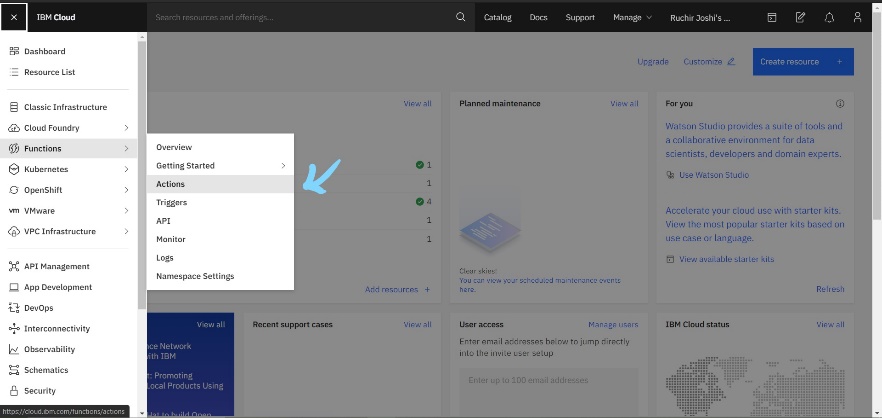


* The URL and API key can be found in the Service credentials tab in the main panel of Discovery service.

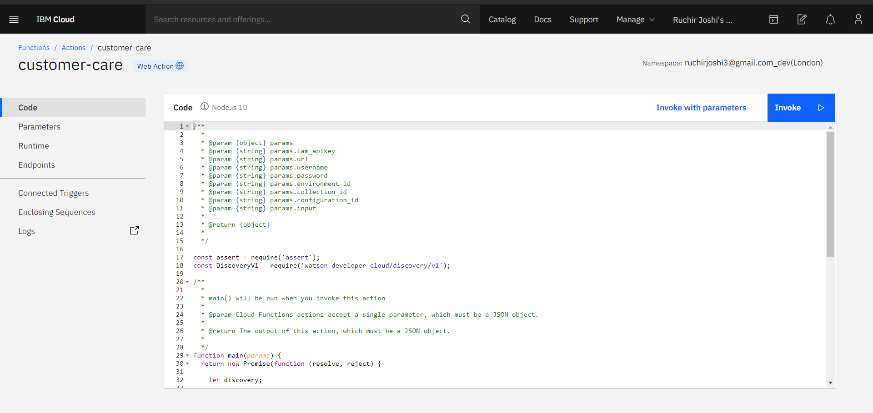


1. Create IBM Cloud Functions action:

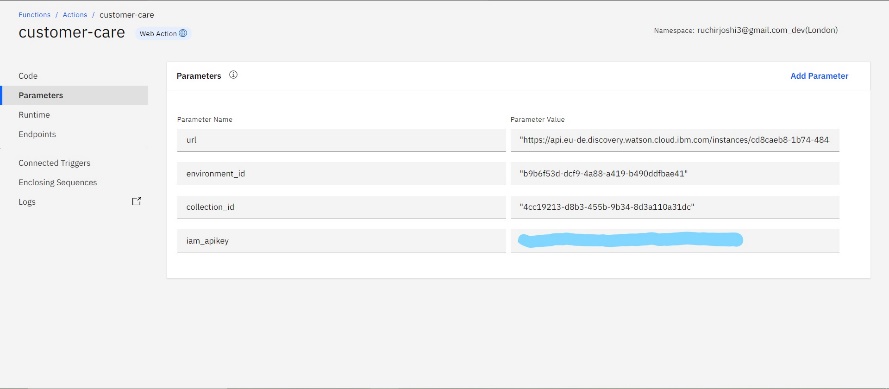
* To create the web action which will make queries against our Discovery collection start the Actions in IBM Cloud Functions service on the main dashboard in the following way.



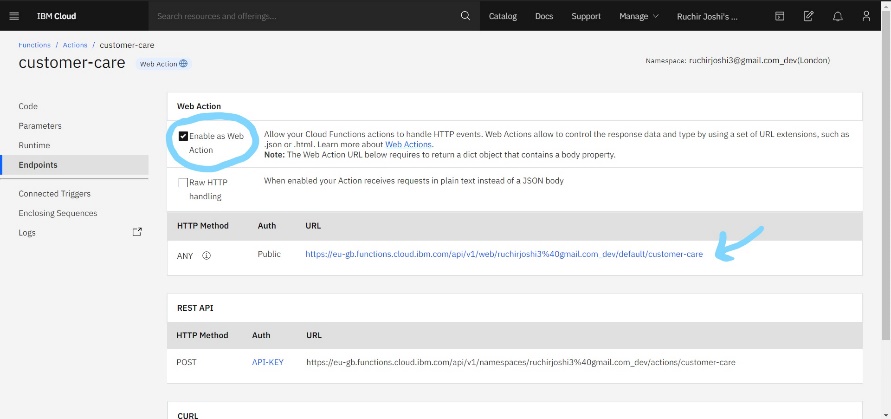
* In the code editor window, paste in the actions code (see Appendix A). This code connects to the Discovery service, makes a query against the collection, then returns the response.



* Adding the following keys url, environment\_id, collection\_id, iam\_apikey in the parameters section and their values are the same from discovery credentials in the above section.



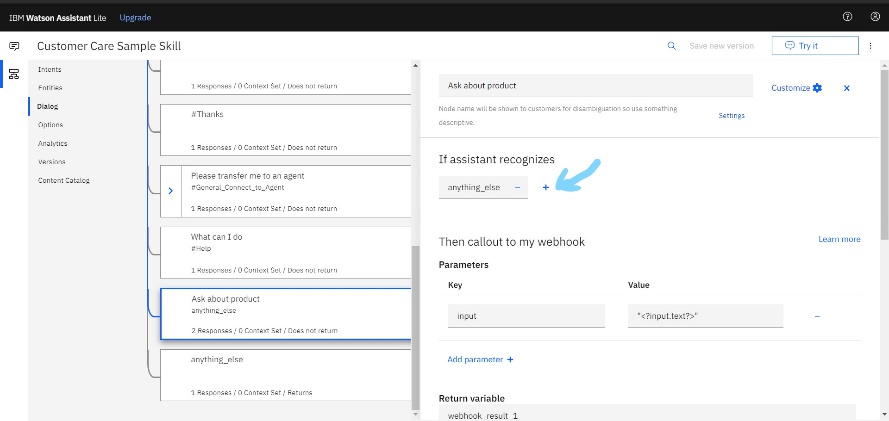
* In the Endpoints tab click the checkbox to Enable as Web Action. This will generate a public endpoint URL. The URL will be needed by the Watson Assistant in the upcoming step



1. Configure Watson Assistant:

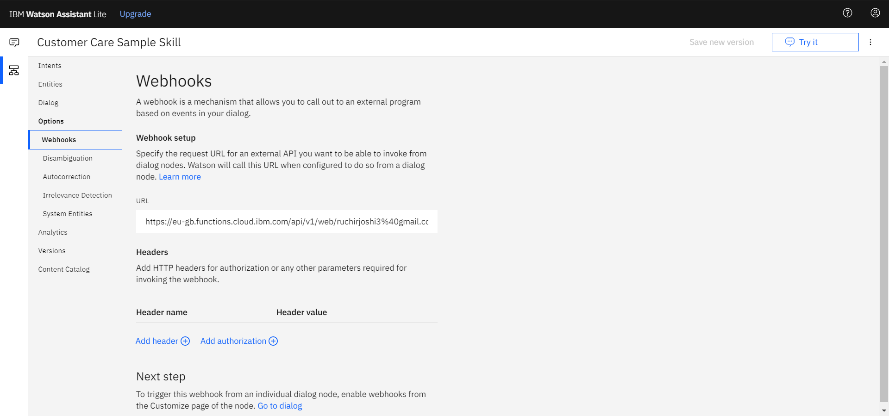
* Create the Watson Assistant tool and create a new dialog skill. Select the Use sample skill option as the starting point. This dialog skill contains all of the nodes needed to have a typical call centre conversation with a user.
* *Create new dialog node*

Naming the node "Ask about product" and assigning it anything\_else option. This means that if Watson Assistant recognizes a user input which are out of its scope, it will direct the conversation to this node.

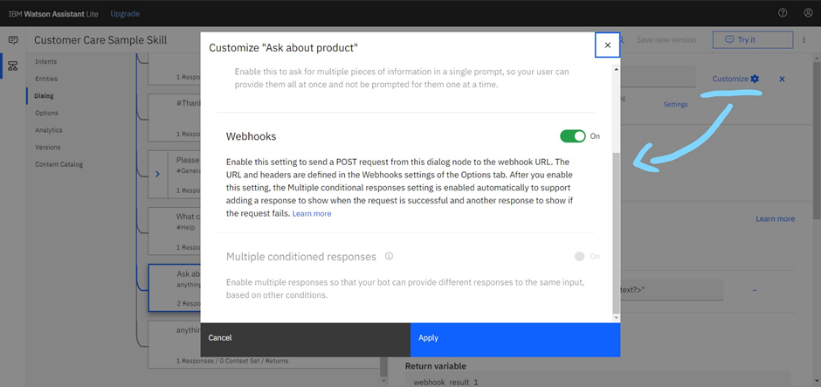


* *Enable webhook from Assistant*

Setting up access to our Webhook for the IBM Cloud Functions action created.



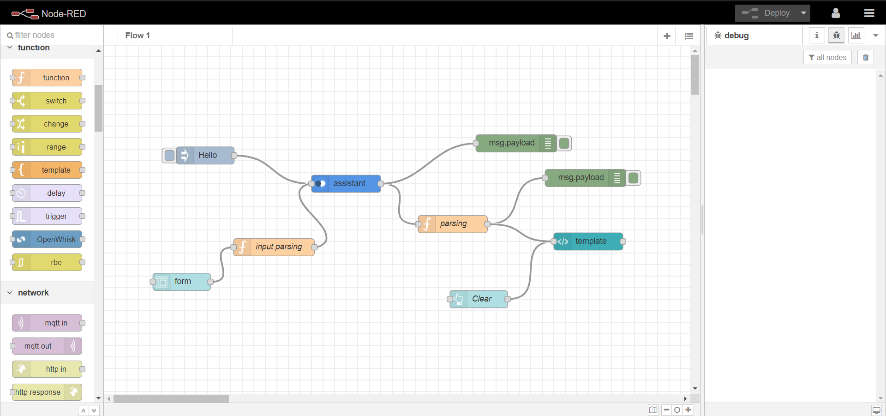
* Enter the public URL endpoint for your IBM Cloud function action. Return to the Dialog tab, and click on the Ask about product node. From the details panel for the node, click on Customize, and enable Webhooks for this node.



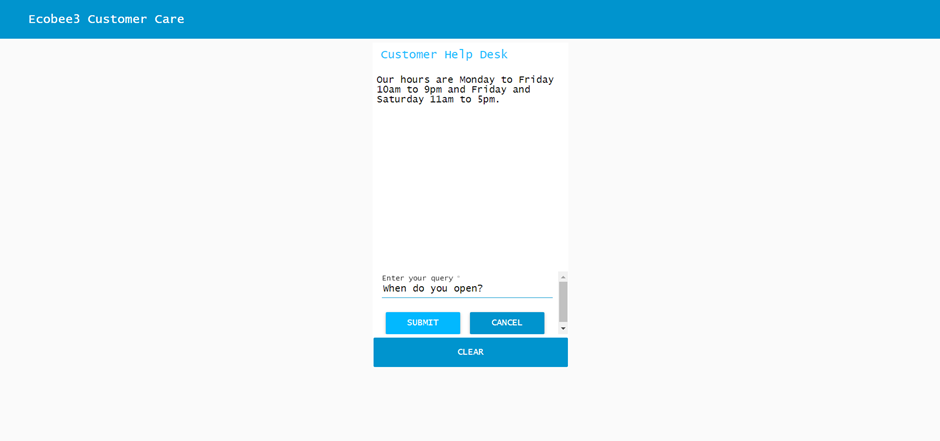
* The dialog node should have a Return variable set automatically to $webhook\_result\_1. This is the variable name which is used to access the result from the Discovery service query.

1. Create flow and configure node:

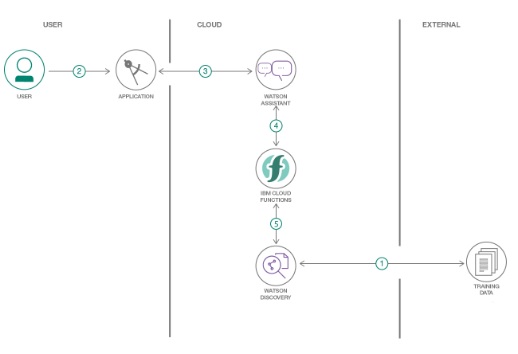
* At first go to manage palette and install dashboard. Create the flow as shown below.



* Deploy the Node Red flow.

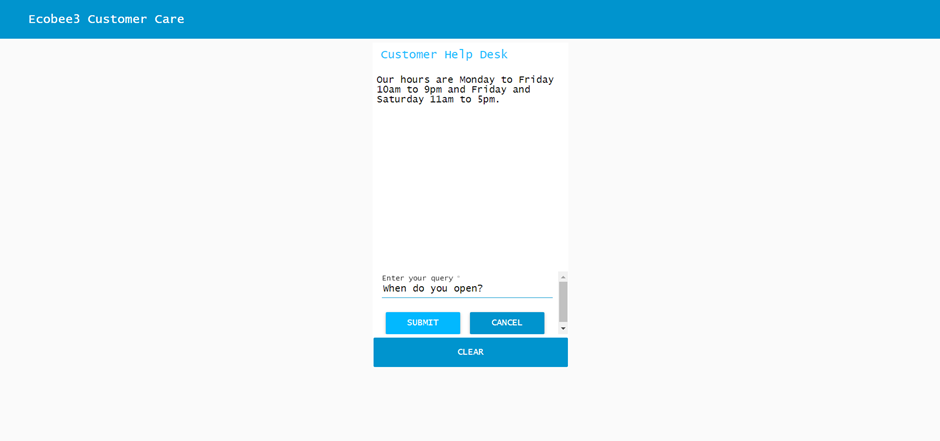


1. **FLOWCHART**



1. **RESULT**

This is the Customer Care UI of the product where the user asks the queries and assistant answer them.



1. **ADVANTAGES & DISADVANTAGES**
   1. **Advantages**

* Reduced costs: Chatbots eliminate the need for labour during online interaction with customers. This is obviously a great advantage for companies that receive multiple queries at once.
* 24/7 Availability: Once we install a chatbot, it can handle queries at any time of day. Thus, the customer does not have to wait for a someone of the company to help him.
* Learning and updating: AI-based chatbots are able to learn from interactions and update independently.
* Management of multiple clients: Humans can serve a limited number of customers at the same time. This restriction does not exist for chatbots, and they can manage all the necessary queries simultaneously.
  1. **Disadvantages**
* Complex interface: It is often considered that chatbots are complicated and need a lot of time to understand what you want in customer. Sometimes, it can also annoy the client about their slowness, or their difficulty in filtering responses. If a query doesn’t relate to something you’ve previously taught it, you won’t understand it. This can lead to a frustrated customer and the loss of the sale.
* Bad memory: The chatbots are not able to memorize a conversation already had, which forces the user to write the same thing over and over again. This can be cumbersome for the client and annoying for the effort required.

1. **APPLICATIONS**

* Help User: This chatbot will be useful for the user to ask the assistant the queries related to the Product and will give them clear guidance about the Product.
* Content delivery: Media Publishers have realized that chatbots are a powerful way to engage with their audiences and monitor engagement to gain valuable insights on reader interests. Chat with the CNN and Wall Street Journal Chatbots on Facebook Messenger and receive the latest news directly in Messenger, without having to visit their websites.
* Companionship: To speak with senior people on general topics like the weather, nature, hobbies, movies, music, news, etc. The chatbot asks questions, reacts to the answers, is able to speak on various topics, and share interesting news and facts from Google.

1. **CONCLUSION**

The rise in popularity of chatbots is clearly known. With this in mind the data gathered from testing the chatbot justifies the recent growth and demand for companies wanting to integrate a chatbot. It was determined that chatbots perform at a very high standard and provide reliable and rapid responses to users compared to that of traditional methods. The average time spent interacting with the chatbot is very low as it provides an efficient way for users to manage their banking. The low interaction time reflects the high understanding and speech recognition rates, offered through the adoption of conversational user interfaces thus allowing users to freely interact with the chatbot to meet the demands of modern life. The chatbot has proven to fulfil the demand of users wanting instant access and availability information and services.

1. **FUTURE SCOPE**

* Giving Chatbots human level understanding and memorization techniques to flawlessly execute chat between human and machines.
* Integration with IoT Devices as Car speakers, smart home devices, and wearables as a few examples where the virtual assistant is departing from its original hardware and making its way to in-context devices
* Voice recognition can be added with the virtual assistant. Then the customer can control application by using his voice

1. **BIBILOGRAPHY**
2. <https://cloud.ibm.com/docs/overview?topic=overview-whatis-platform>
3. <https://www.ibm.com/cloud/architecture/tutorials/cognitive_discovery>
4. <https://nodered.org/>
5. <https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/>
6. <https://www.ibm.com/cloud/watson-assistant/>
7. <https://cloud.ibm.com/docs/openwhisk?topic=openwhisk-getting-started>
8. <http://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red>
9. **APPENDIX**
10. **Source code**

/\*\*

\* @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': '2020-05-20'

});

}

else {

discovery = new DiscoveryV1({

'username': params.username,

'password': params.password,

'url': params.url,

'version': '2020-05-20'

});

}

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);

});

});

}