**Intelligent Customer Help Desk With Smart**  **Document Understanding**

**1. Introduction**

**1.1 Overview**

Intelligent Customer Help Desk With Smart Document Understanding is a project in which a bot is created which answers typical customer's questions such as store locations,hours and even take appointments. This bot can also answer questions which are out of the scope rather than tranferring the call to representatives.

It is made using IBM Watson Assistant and uses IBM Watson Discovery Service to answer out of scope questions.

**1.2 Purpose**

The purpose of this project is to provide customers with end to end user experience without need of any customer support staff. They can ask queries related to product and the bot will communicate behaving like a human. The bot is available 24/7 and it can cater huge amount of audience at the same time.And this bot gets trained itself based on the questions asked by the customers. So, basically, it reduces the need of working staff for the company.

**2. Literature Survey**

**2.1 Existing problem**

Currently for customer helpdesk we have representatives available to solve queries of customers. But to provide customer support to large number of users at the same time and to be available 24/7 is a difficult task. Moreover there are bots present but if a customer asks a question out of scope the bot transfers the calls to representatives i.e they are unable to solve the queries out of scope.

**2.2 Proposed solution**

The solution provided here is a bot made using IBM Watson Assistant which is able to cater a lot of customers at the same time and it is available 24/7 and if for the questions which are out of scope it used IBM Watson Discovery Service . This service trains the bot based on the dataset provided by the company like user manual. Also, using Smart Document Understanding feature of Watson Discovery the bot is trained on what text is important from User's manual and what is not.

**3. Theoretical Analysis**

**3.1 Block Diagram**

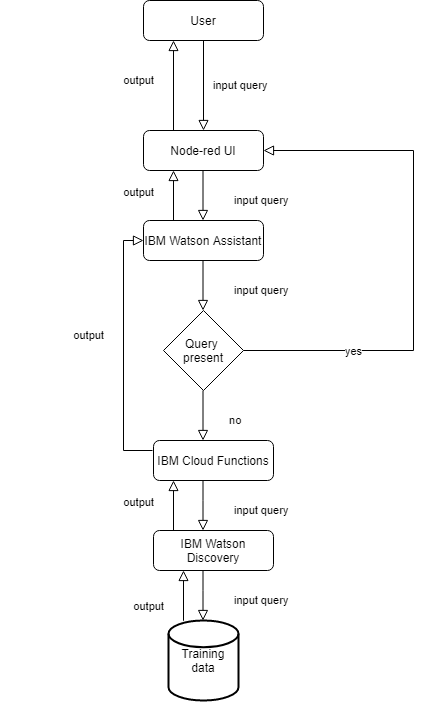
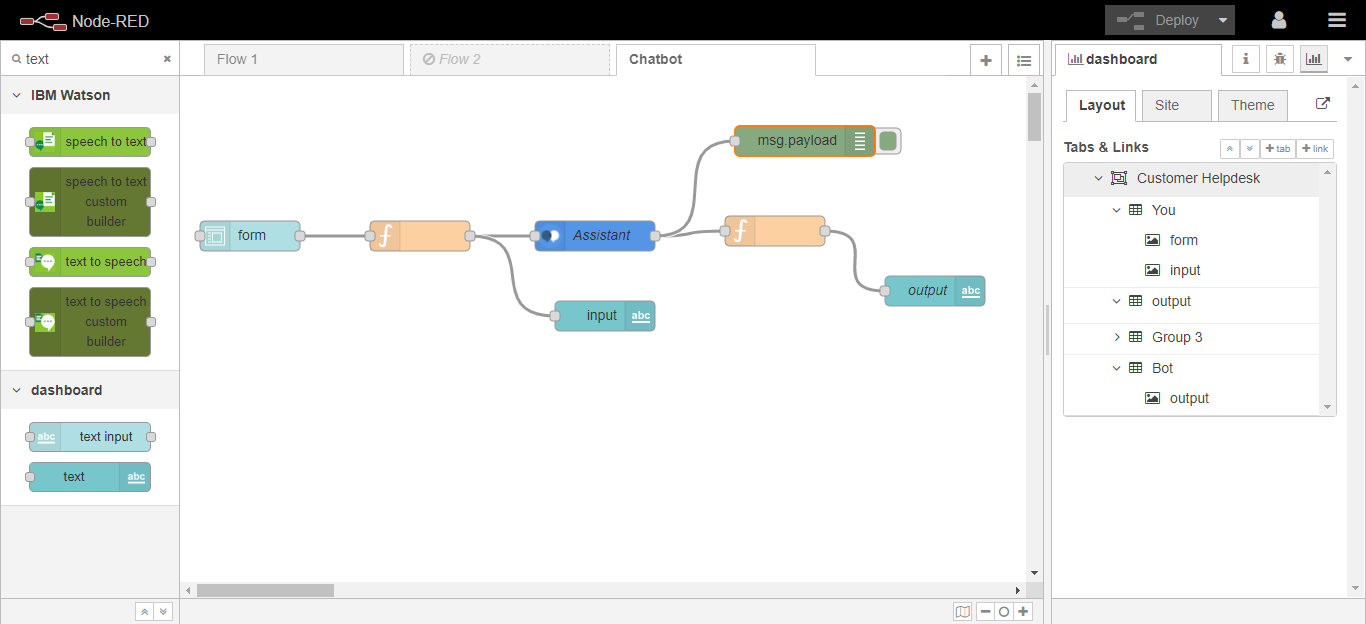


Fig 1. Block diagram

**3.2 Hardware/Software Designing**

Services used are IBM Watson Assistant , IBM Watson Discovery, IBM Cloud Functions and Node-red to integrate them and deploy by creating a flow.Proper credentials were entered in Assistant node and the connections were added and the input and output messages were configured using javascript functions and they were represented properly using UI to the end user . The Flow of node-red is as below:



**4. Experimental Investigations**

1. Create IBM Services

* Watson Assistant
* Watson Discovery

2. Configure Watson Discovery

* Import the document
* Annotate with Smart Document Understanding
* Store credentials for future use

3. Create IBM Cloud Functions

* Create new Action
* Add the following keys in parameters:
  + url
  + environment\_id
  + collection\_id
  + iam\_apikey
* Use values associated with Discovery Service
* Enable Web Action to get an endpoint

4. Configure Watson Assistant

* Use sample skill option
* Add new INTENT
* Create new DIALOG node
* Enable webhook from Assistant
* Enter the url endpoint of cloud function and add .json at the end
* Enable webhook for newly created DIALOG node
* Add input parameter in Callout to Webhook
* Test in Assistant tooling

5. Build Node-red flow to integrate all services

* Create the required nodes and connect them accordingly
* Enter url, workspace\_id and apikey to ASSISTANT node
* Modify the UI and Deploy
* Test the Bot and capture the results

**5. Flowchart**

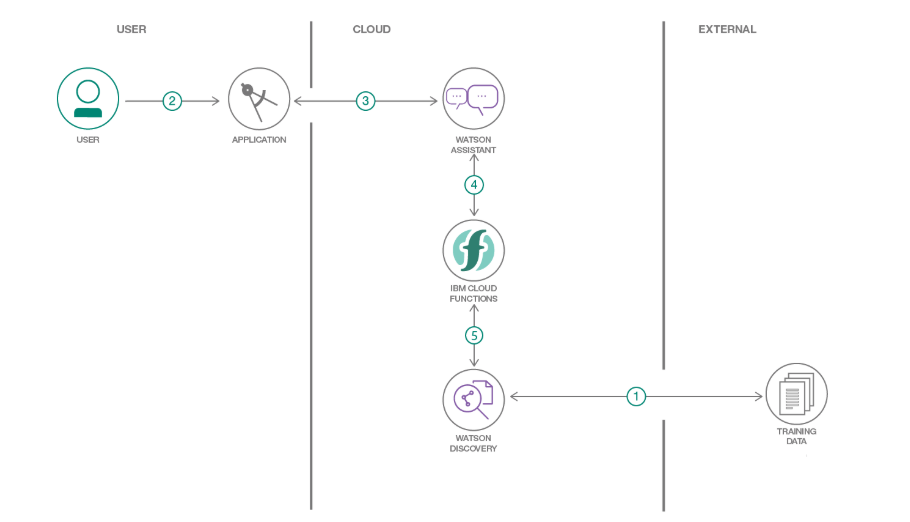


Fig 2. Flowchart

1. The document is annotated using Watson Discovery SDU
2. The user interacts with the backend server via the node-red UI. The front end app UI is a chatbot that engages the user in a conversation.
3. Dialog between the user and backend server is coordinated using a Watson Assistant dialog skill.
4. If the user asks a product operation question, a search query is passed to a predefined IBM Cloud Functions action.
5. The Cloud Functions action will query the Watson Discovery service and return the results.

**6. Result**

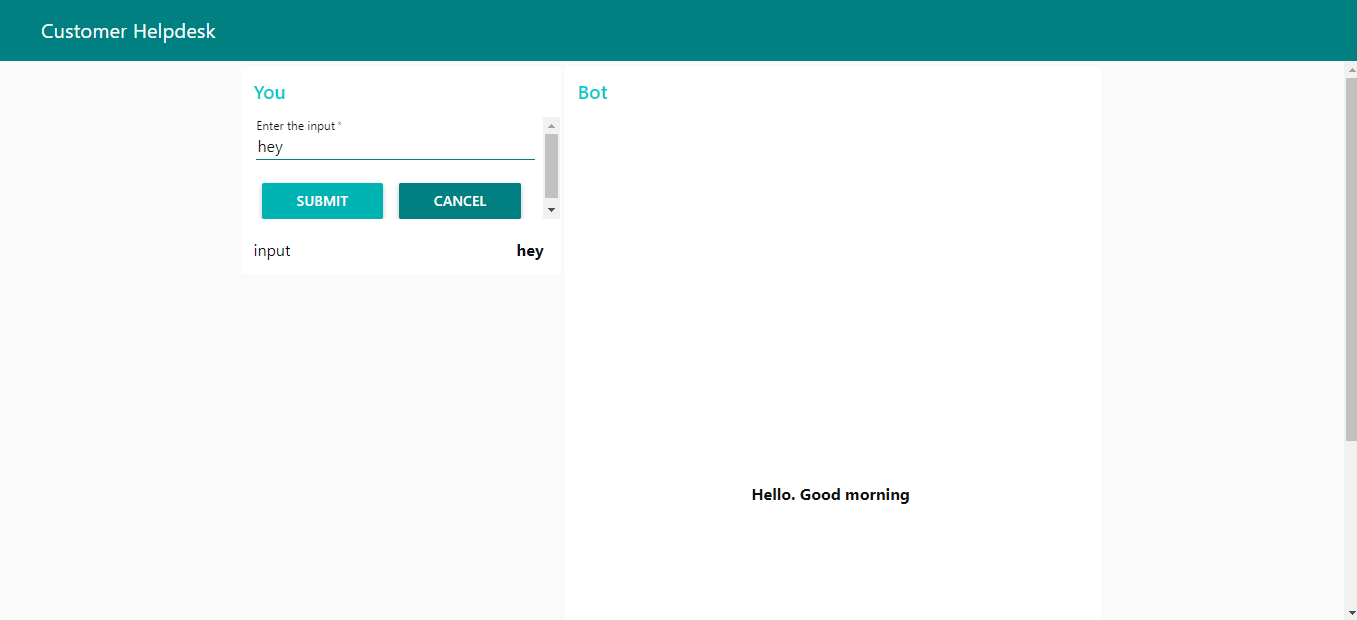


Fig 3. Greetings

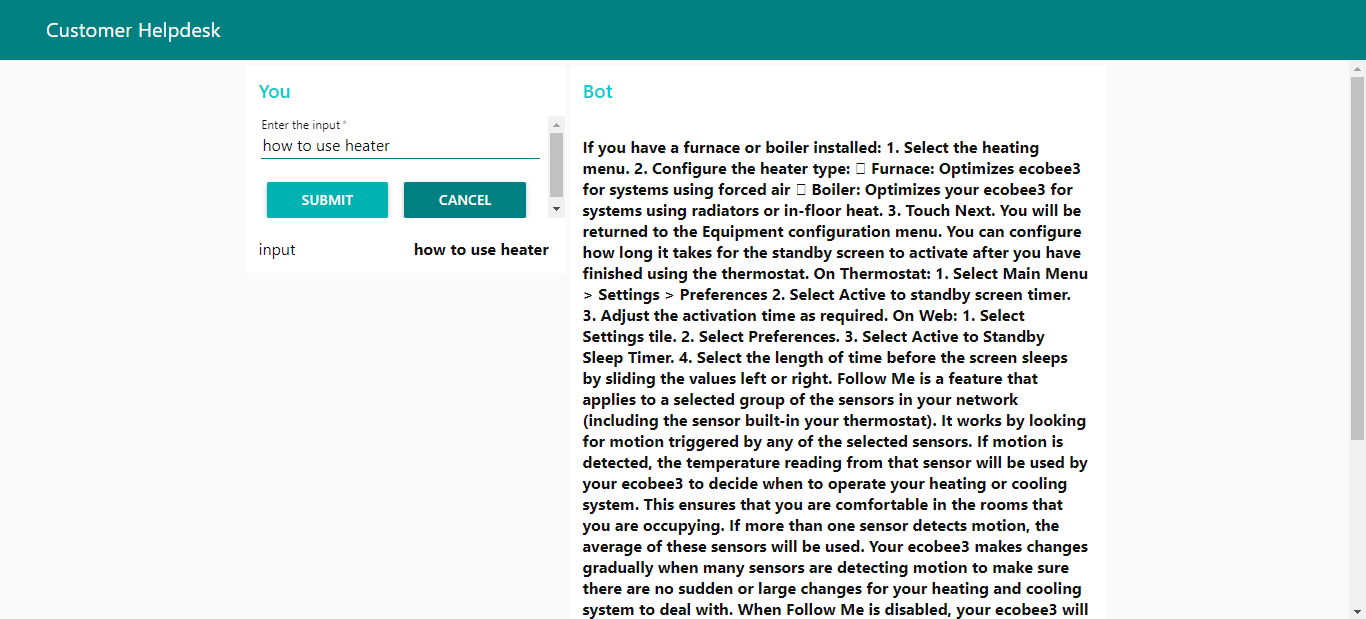


Fig 4(1). Product Information

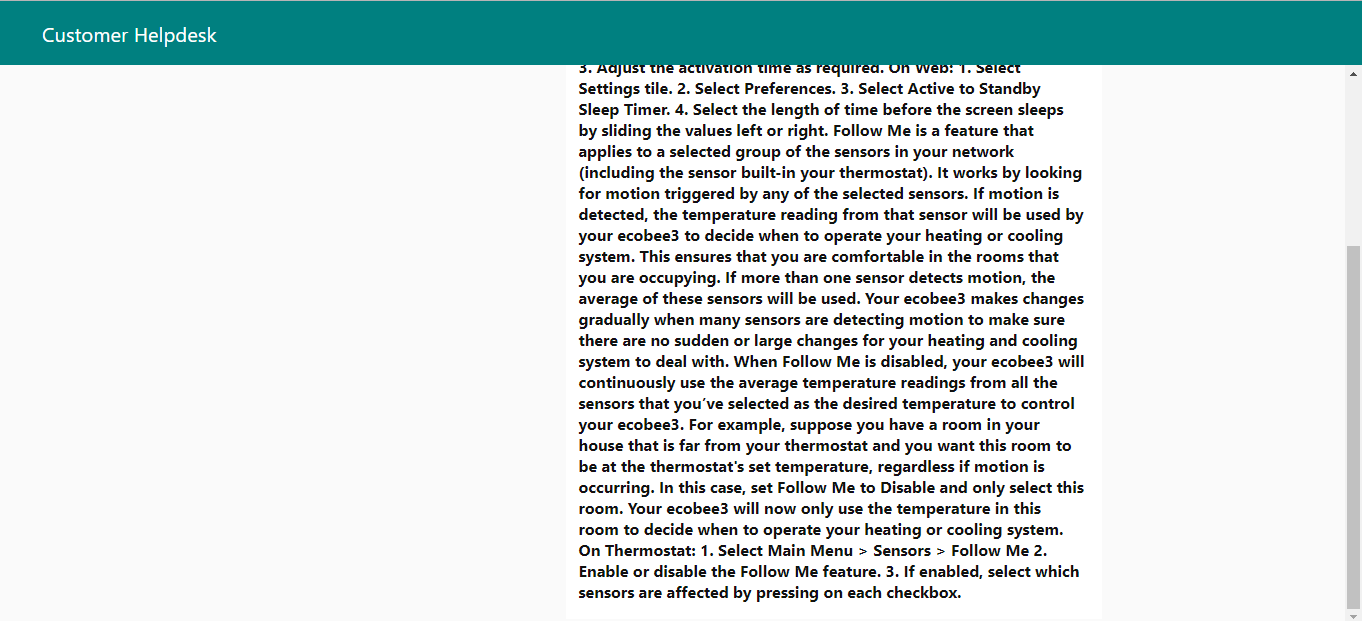


Fig 4(2). Product Information

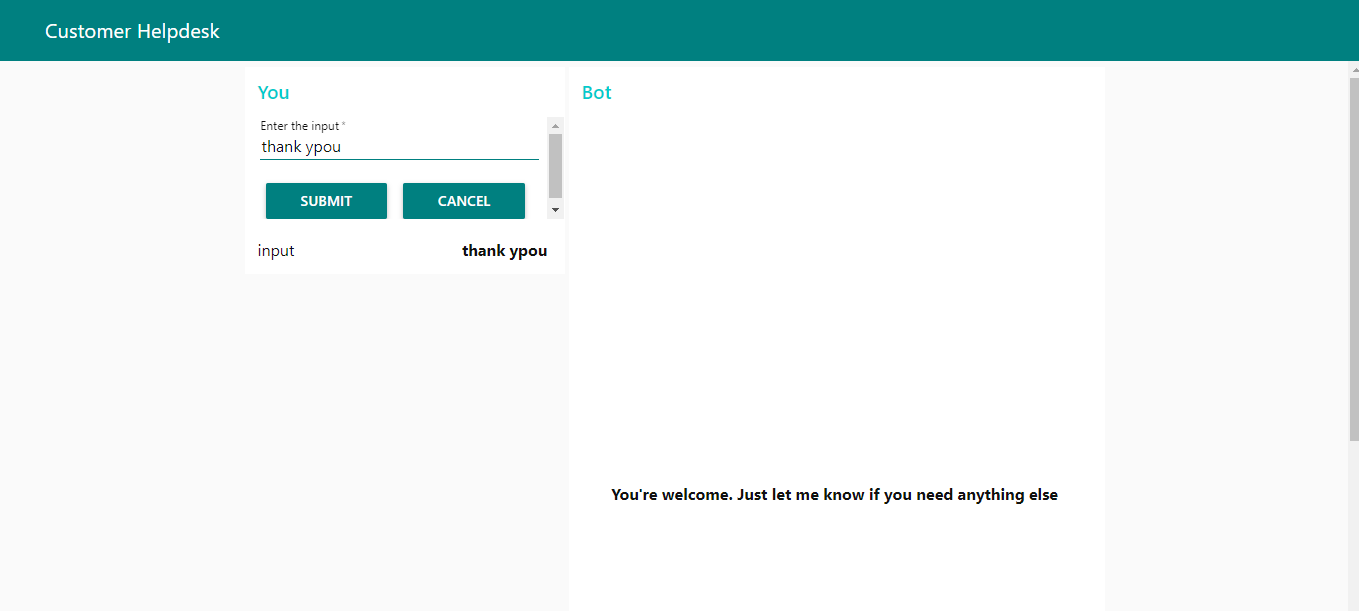


Fig 5. Thanks

**7. Advantages & Disadvantages**

**7.1 Advantages**

* Can answer typical Customer care questions such as store locations, hours, schedule appointments.
* Out of scope questions are also handled so no need to call customer care representative for such questions.
* No requirement of human for answering queries.
* Can handle large amount of user traffic at the same time.
* Available 24/7 , so customer can use it anytime to solve their queries.

**7.2 Disadvantages**

* Good internet connection is required.
* May not answer all type of customer queries.

**8. Applications**

* E-commerce
* Content delivery
* Event reservation
* Handling minor but important tasks
* Personalized helpers
* Travel or conference bots

**9. Conclusion**

Through this project , the conclusion can be drawn that the bot is very useful in solving typical as well as out of the scope queries of the customers. And with the use of IBM Discovery and Smart Document Understanding it also overcomes the major flaws which are present in all the chatbots which are currently used. No human intervention is needed and it can work 24/7 .

**10. Future Scope**

With the use of Smart Document Understanding the bot can be trained on more than one dataset .It can be integrated with various things like Google Maps to provide navigation information, WhatsApp, Facebook,etc . It can have audio recording option so that users queries can be recorded and then sent to representatives so the user dont have to wait if no representatives are available.

**11. Bibliography**

* <https://www.ibm.com/cloud/get-started>
* <https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/>
* <https://nodered.org/>
* <https://developer.ibm.com/components/watson-assistant/series/learning-path-watson-assistant>
* <https://developer.ibm.com/articles/introduction-watson-discovery/>
* <https://cloud.ibm.com/docs/services/discovery?topic=discovery-getting-started>
* <https://cloud.ibm.com/docs/openwhisk?topic=cloud-functions-getting-started>

**Appendix**

**A. Source code**

**flows.json**

[

{

"id": "704c1194.e2dac",

"type": "tab",

"label": "Chatbot",

"disabled": false,

"info": ""

},

{

"id": "7247f5a7.13aeec",

"type": "watson-conversation-v1",

"z": "704c1194.e2dac",

"name": "Assistant",

"workspaceid": "cfb77e7c-cae6-49d5-8ebc-97801042aa49",

"multiuser": false,

"context": false,

"empty-payload": false,

"service-endpoint": "https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/63d20861-f832-44e4-b953-43cfbc2d1776",

"timeout": "",

"optout-learning": false,

"x": 415,

"y": 160,

"wires": [

[

"9a8fdec8.4daca",

"f5001686.7c8c68"

]

]

},

{

"id": "2cb7553e.b5707a",

"type": "function",

"z": "704c1194.e2dac",

"name": "",

"func": "msg.payload = msg.payload.input;\nreturn msg;",

"outputs": 1,

"noerr": 0,

"x": 240,

"y": 160,

"wires": [

[

"7247f5a7.13aeec",

"c299e726.e3fc48"

]

]

},

{

"id": "9a8fdec8.4daca",

"type": "function",

"z": "704c1194.e2dac",

"name": "",

"func": "msg.payload.text=\"\";\nif(msg.payload.context.webhook\_result\_1){\n for(var i in msg.payload.context.webhook\_result\_1.results){\n msg.payload.text=msg.payload.text+\"\\n\"+msg.payload.context.webhook\_result\_1.results[i].text;\n }\n msg.payload=msg.payload.text;\n}\nelse{\n msg.payload=msg.payload.output.text[0];\n}\nreturn msg;",

"outputs": 1,

"noerr": 0,

"x": 595,

"y": 155,

"wires": [

[

"bd084ba5.525fe8"

]

]

},

{

"id": "f5001686.7c8c68",

"type": "debug",

"z": "704c1194.e2dac",

"name": "",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "false",

"x": 625,

"y": 65,

"wires": [

]

},

{

"id": "bd084ba5.525fe8",

"type": "ui\_text",

"z": "704c1194.e2dac",

"group": "9b3d6a0e.17d638",

"order": 4,

"width": "10",

"height": "14",

"name": "output",

"label": "",

"format": "{{msg.payload}}",

"layout": "col-center",

"x": 755,

"y": 215,

"wires": [

]

},

{

"id": "a6348aaf.80a558",

"type": "ui\_form",

"z": "704c1194.e2dac",

"name": "",

"label": "",

"group": "6ce32c09.3f5f54",

"order": 2,

"width": 0,

"height": 0,

"options": [

{

"label": "Enter the input",

"value": "input",

"type": "text",

"required": true,

"rows": null

}

],

"formValue": {

"input": ""

},

"payload": "",

"submit": "submit",

"cancel": "cancel",

"topic": "",

"x": 70,

"y": 160,

"wires": [

[

"2cb7553e.b5707a"

]

]

},

{

"id": "c299e726.e3fc48",

"type": "ui\_text",

"z": "704c1194.e2dac",

"group": "6ce32c09.3f5f54",

"order": 4,

"width": 0,

"height": 0,

"name": "",

"label": "input",

"format": "{{msg.payload}}",

"layout": "row-spread",

"x": 425,

"y": 240,

"wires": [

]

},

{

"id": "9b3d6a0e.17d638",

"type": "ui\_group",

"z": "",

"name": "Bot",

"tab": "ff734b8.ae7cfb8",

"order": 4,

"disp": true,

"width": "10",

"collapse": false

},

{

"id": "6ce32c09.3f5f54",

"type": "ui\_group",

"z": "",

"name": "You",

"tab": "ff734b8.ae7cfb8",

"order": 1,

"disp": true,

"width": "6",

"collapse": false

},

{

"id": "ff734b8.ae7cfb8",

"type": "ui\_tab",

"z": "",

"name": "Customer Helpdesk",

"icon": "dashboard",

"disabled": false,

"hidden": false

}

]

**cloudfunctions.js**

/\*\*  
 \*  
 \* @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);  
 });  
 });  
}