**Intelligent Customer Help**

**Desk with Smart Document**

**Understanding**

**by**

**Ashish Manghani**

**1. Introduction..................................................................** 3

1.1 Overview**................................................................................................** 3

1.2 Purpose**..................................................................................................** 4

**2. Literature Survey.........................................................** 5

2.1 Existing Problem**....................................................................................** 5

2.2 Proposed Solution**.................................................................................** 5

**3. Theoretical Analysis....................................................** 6

3.1 Block Diagram**........................................................................................** 6

3.2 Hardware / Software Designing**.............................................................** 6

**4. Experimental Investigations.......................................** 7

**5. Flowchart......................................................................** 9

**6. Result..........................................................................** 10

**7. Advantages & Disadvantages...................................** 11

**8. Applications................................................................** 12

**9. Conclusion..................................................................** 13

**10. Future Scope............................................................** 14

**11. Bibliography.............................................................** 15

**Appendix........................................................................** 16

A. Source Code**........................................................................................** 16

**1. Introduction**

* 1. **Overview**
* We will build a chatbot that uses various IBM Watson AI Services (Watson Discovery, Watson Assistant, Watson Cloud Functions and Node-Red) to deliver an effective Web based UI through which we can chat with the assistant.
* We will integrate the Watson Discovery service with Watson Assistant

using webhooks.

* Project Requirements: Node-RED, IBM Cloud, IBM Watson, Node JS
* Functional Requirements: The user must be able to query and get information on the topics which are part of the documententation used by Watson Discovery.

The service should also work properly for the regular queries asked to the chatbot.

* Technical Requirements: AI, ML, Watson AI, Node JS
* Software Requirements: Watson Assistant, Watson Discovery,

Watson Cloud Functions, Node-RED

* Project Deliverables: Intelligent Chatbot with Smart Document

Understanding

* Project Duration: 19 Days
  1. **Purpose**

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. 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. So now, instead of “Would you like to speak to a customer representative?” we can return relevant sections of the owner manual to help solve our customers’ problems. So unless and until customer specifically asks for a customer representative the bot will try to solve all your queries.

To take it a step further, the project shall use 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. Then using Watson actions as webhook, Watson Discovery can be integrated with Watson assistant. Finally using Node-Red, Watson assistant can be integrated with a web UI. This UI can then be used to connect with Watson assistant and chat with it.

**1.2.1 Scope of Work**

* Create a customer care dialog skill in Watson Assistant
* Use Smart Document Understanding to build an enhanced Watson

Discovery collection

* Create an IBM Cloud Functions web action that allows Watson

Assistant to post queries to Watson Discovery

* Build a web application with integration to all these services & deploy

the same on IBM Cloud Platform

**2. Literature Survey**

**2.1 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

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.

**2.2 Proposed Solution**

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. So now, instead of “Would you like to speak to a

customer representative?” we can return relevant sections of the owners

manual to help solve our customers’ problems. So unless and untill

customer specifically asks for a customer representative the bot will try to

solve all your queries.

To take it a step further, the project shall use 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

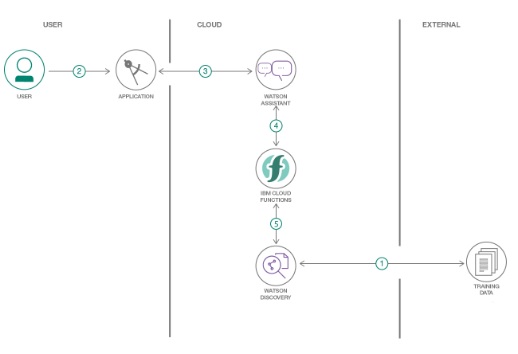
returned from the queries. Then using Watson actions as webhook, Watson

Discovery can be integrated with Watson assistant. Finally using Node-Red,

Watson assistant can be integrated with a web UI. This UI can then be used

to connect with Watson assistant and chat with it.

**3. Theoretical Analysis**



Block / Flow Diagram

Hardware / Software Designing

1. Create necessary Watson Services.

2. Configure Watson Discovery.

3. Create Watson Cloud Functions Action.

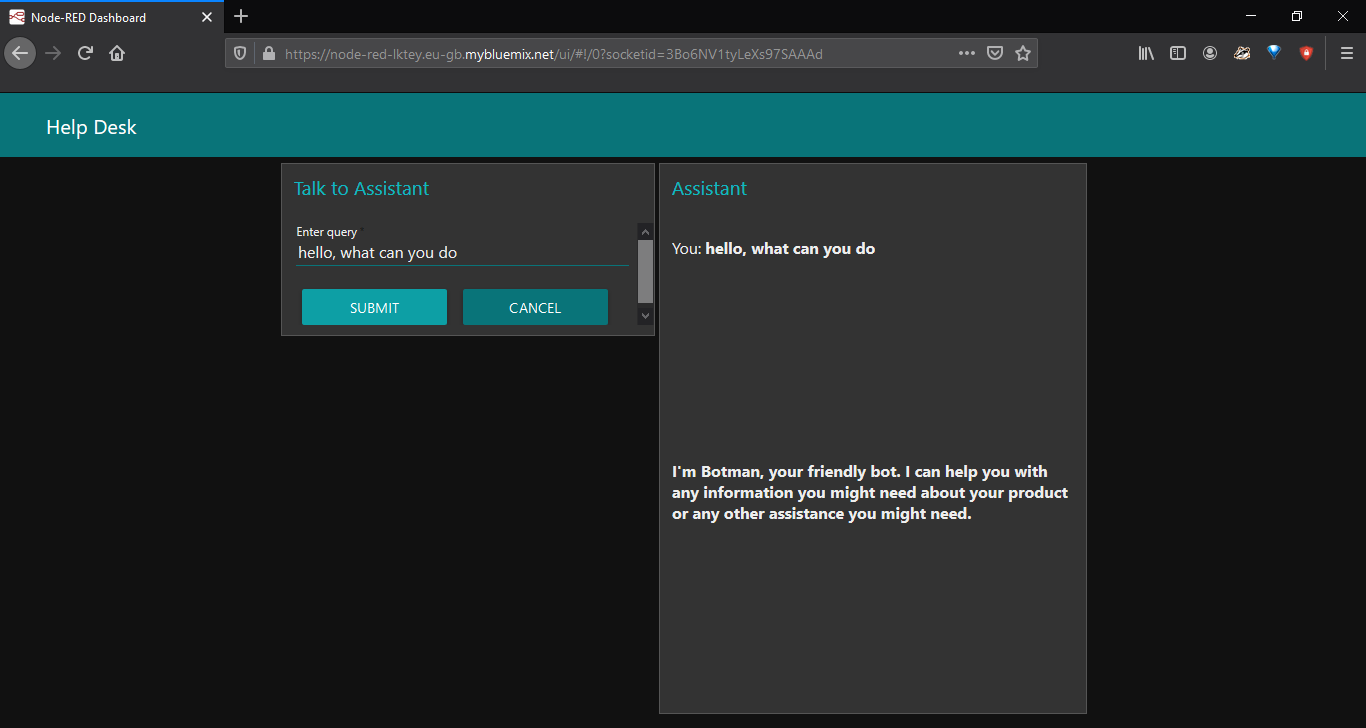
4. Configure Watson Assistant.

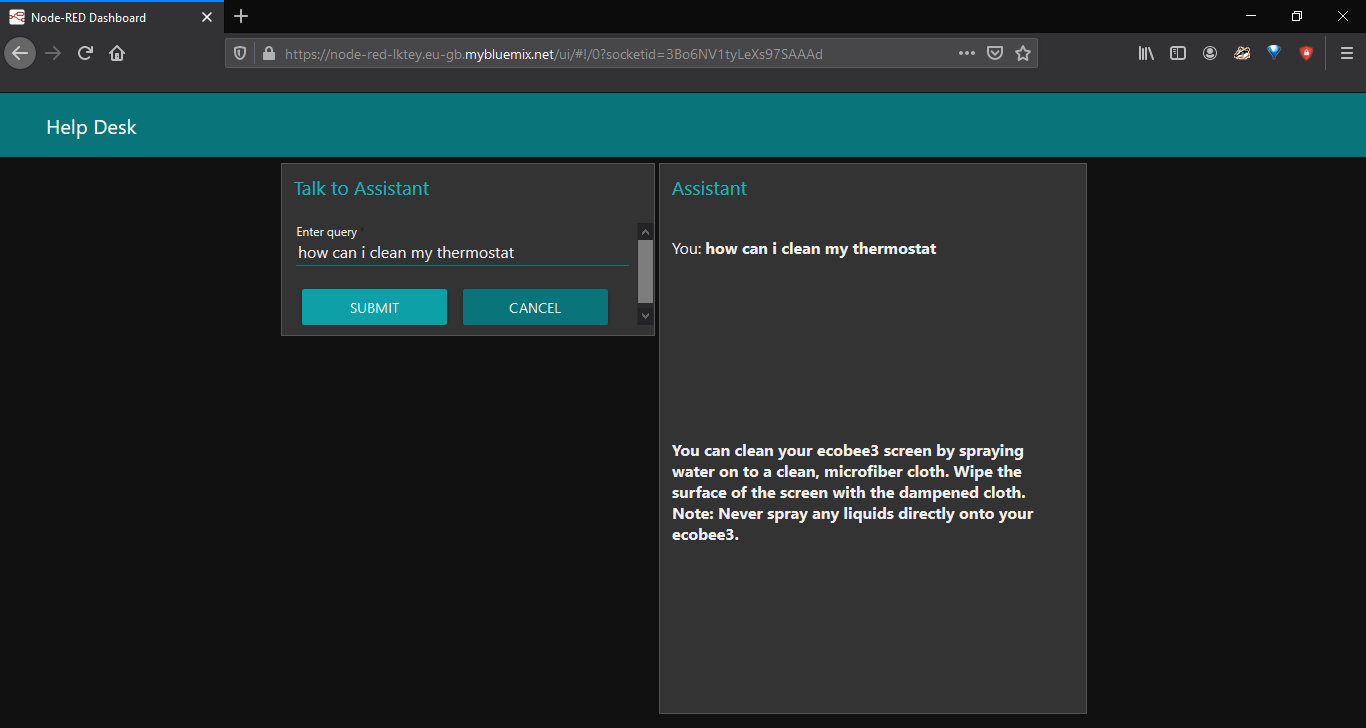
5. Integrate Watson Discovery with Watson Assistant using webhook.

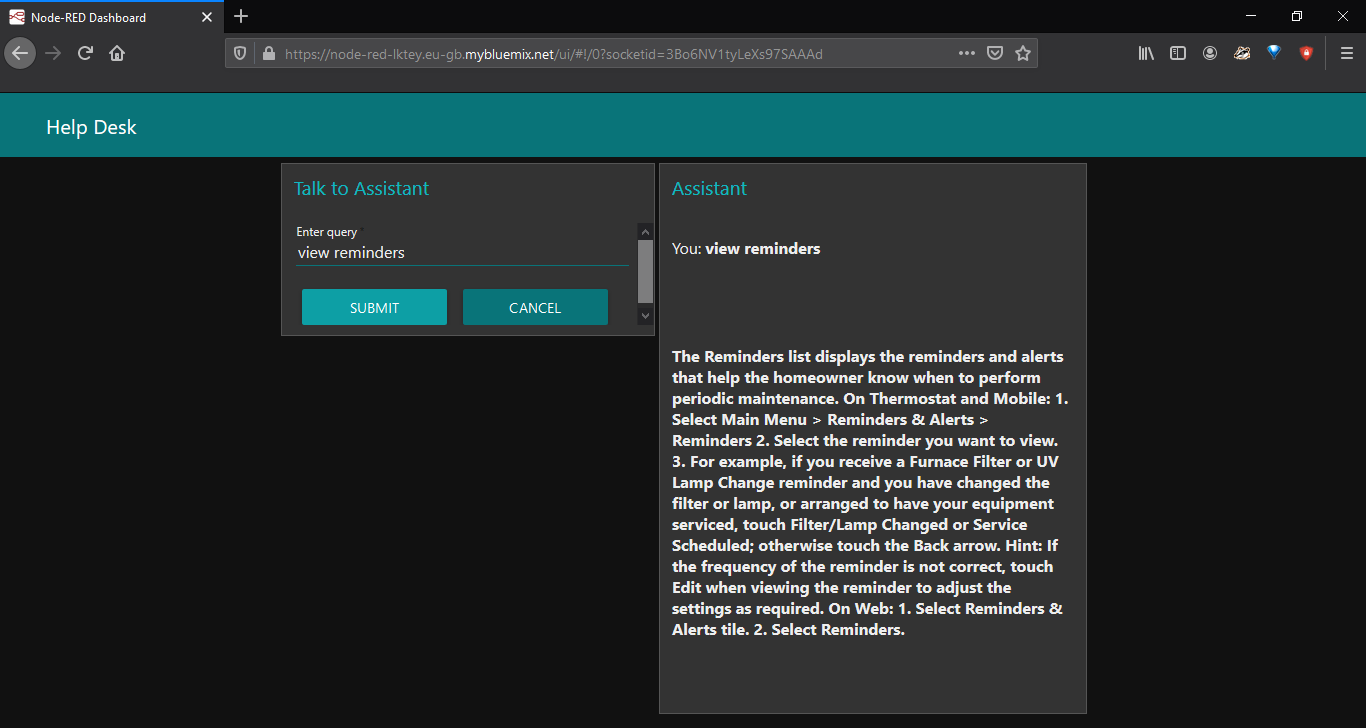
6. Build Node-RED flow to integrate Watson Assistant and Web

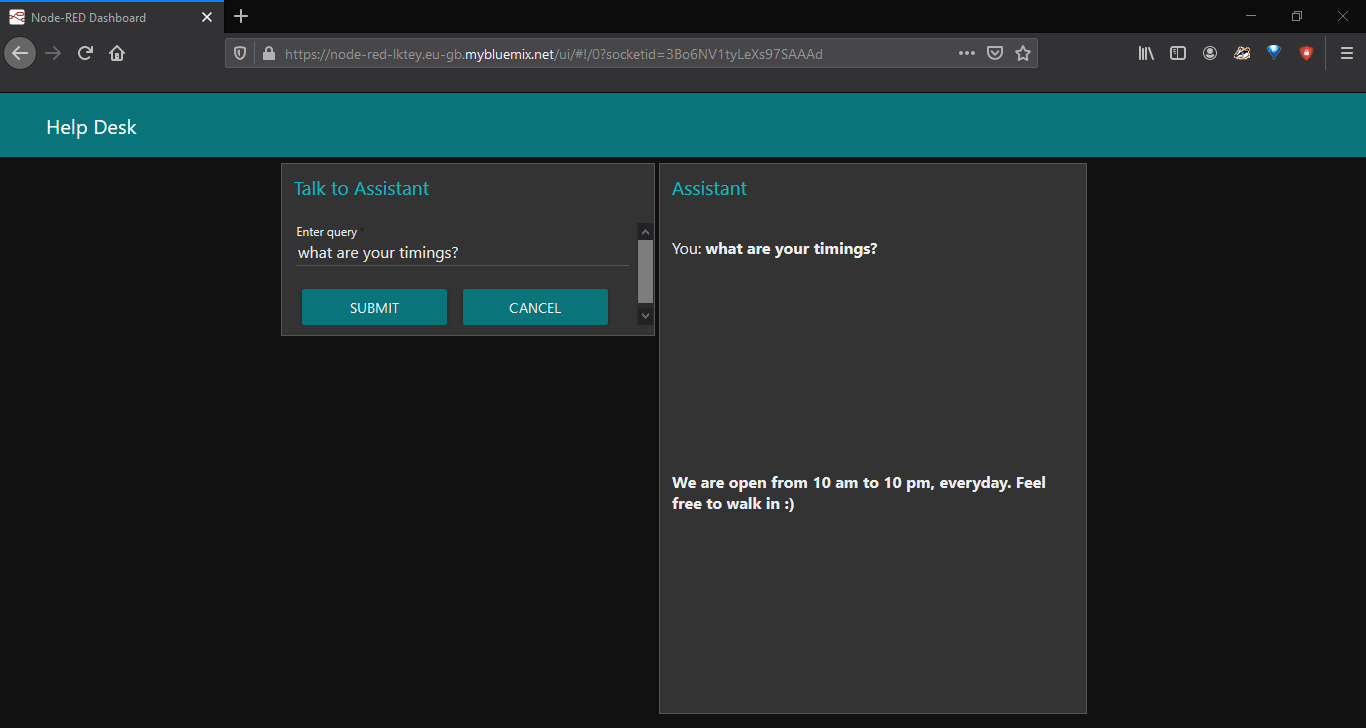
Dashboard.

**4. Experimental Investigation**



****

****

****

**5. Flowchart**

Insert the following nodes into the flow in Node-RED.

* UI Form
* Function
* UI Text
* Assistant
* Debug
* Function
* Debug
* UI Text

**6. Results**

Web based UI for chatbot was developed by integrating all the services using

Node-RED.

URL for UI Dashboard: https://node-red-lktey.eu-gb.mybluemix.net/ui

**7. Advantages & Disadvantages**

**Advantages**

1. Reduces Man Power

2. Cost Efficient

3. Less and less calls will be diverted to Customer Representatives.

**Disadvantages**

1. Sometimes it can mislead customers as it tries to search irrelevant information

in the manual.

2. It may also give same answers to different queries.

3. The results from documentation might not be very clear for the customer to understand.

**8. Applications**

1. This chatbot can be deployed to various websites as it can solve a lot of basic

questions.

2. It can be used to deploy as Customer Helpdesk for small scale products as

their manual usually has the solution for the user's problems.

**9. Conclusion**

An Intelligent Customer Helpdesk Chatbot was created using various Watson

services like Watson Discovery, Watson Assistant, Watson Cloud Functions and

Node-RED.

**10. Future Scope**

In the future, various other Watson services like Speech-To-Text can be integrated in the chatbot. This can make the chatbot hands-free. We can also ask the user inputs on chatbot to further improve it.

**11. Bibliography**

1. Node-RED Starter Application:

https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-applicat

ion/

2. Build your own AI assistant:

https://www.youtube.com/watch?v=hitUOFNne14

3. How to use Watson Assistant with Webhooks:

https://www.youtube.com/embed/5z3i5IsBVnk

4. Watson Discovery:

https://developer.ibm.com/articles/introduction-watson-discovery/

**Appendix**

**Source Code**

**Node-RED Flow code**

[

{

"id": "6e79dbc7.7e3c14",

"type": "tab",

"label": "Flow 1",

"disabled": false,

"info": ""

},

{

"id": "e34f18b9.a010d8",

"type": "ui\_tab",

"z": "",

"name": "Help Desk",

"icon": "dashboard",

"disabled": false,

"hidden": false

},

{

"id": "23e816ee.1f29b2",

"type": "ui\_group",

"z": "",

"name": "Talk to Assistant",

"tab": "e34f18b9.a010d8",

"order": 1,

"disp": true,

"width": "7",

"collapse": false

},

{

"id": "ebf67b88.639",

"type": "ui\_base",

"theme": {

"name": "theme-dark",

"lightTheme": {

"default": "#0094CE",

"baseColor": "#0094CE",

"baseFont": "-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",

"edited": true,

"reset": false

},

"darkTheme": {

"default": "#097479",

"baseColor": "#097479",

"baseFont": "-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",

"edited": true,

"reset": false

},

"customTheme": {

"name": "Untitled Theme 1",

"default": "#4B7930",

"baseColor": "#4B7930",

"baseFont": "-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"

},

"themeState": {

"base-color": {

"default": "#097479",

"value": "#097479",

"edited": false

},

"page-titlebar-backgroundColor": {

"value": "#097479",

"edited": false

},

"page-backgroundColor": {

"value": "#111111",

"edited": false

},

"page-sidebar-backgroundColor": {

"value": "#000000",

"edited": false

},

"group-textColor": {

"value": "#0eb8c0",

"edited": false

},

"group-borderColor": {

"value": "#555555",

"edited": false

},

"group-backgroundColor": {

"value": "#333333",

"edited": false

},

"widget-textColor": {

"value": "#eeeeee",

"edited": false

},

"widget-backgroundColor": {

"value": "#097479",

"edited": false

},

"widget-borderColor": {

"value": "#333333",

"edited": false

},

"base-font": {

"value": "-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"

}

},

"angularTheme": {

"primary": "indigo",

"accents": "blue",

"warn": "red",

"background": "grey"

}

},

"site": {

"name": "Node-RED Dashboard",

"hideToolbar": "false",

"allowSwipe": "false",

"lockMenu": "false",

"allowTempTheme": "true",

"dateFormat": "DD/MM/YYYY",

"sizes": {

"sx": 48,

"sy": 48,

"gx": 6,

"gy": 6,

"cx": 6,

"cy": 6,

"px": 0,

"py": 10

}

}

},

{

"id": "e06c378a.d686d",

"type": "ui\_group",

"z": "",

"name": "Assistant",

"tab": "e34f18b9.a010d8",

"order": 2,

"disp": true,

"width": 8,

"collapse": false

},

{

"id": "6a526bb3.ee0aec",

"type": "ui\_spacer",

"name": "spacer",

"group": "e06c378a.d686d",

"order": 2,

"width": 1,

"height": 1

},

{

"id": "f2f2649a.0d0d98",

"type": "debug",

"z": "6e79dbc7.7e3c14",

"name": "",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "payload",

"targetType": "msg",

"x": 730,

"y": 200,

"wires": []

},

{

"id": "124c0e4e.0fb66a",

"type": "ui\_form",

"z": "6e79dbc7.7e3c14",

"name": "",

"label": "",

"group": "23e816ee.1f29b2",

"order": 1,

"width": 7,

"height": "2",

"options": [

{

"label": "Enter query",

"value": "input",

"type": "text",

"required": true,

"rows": null

}

],

"formValue": {

"input": ""

},

"payload": "",

"submit": "submit",

"cancel": "cancel",

"topic": "",

"x": 90,

"y": 320,

"wires": [

[

"3682a84b.0aa51"

]

]

},

{

"id": "3682a84b.0aa51",

"type": "function",

"z": "6e79dbc7.7e3c14",

"name": "",

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

"outputs": 1,

"noerr": 0,

"x": 230,

"y": 240,

"wires": [

[

"cc13951b.ef59b",

"6aed6986.c684f"

]

]

},

{

"id": "3e2c8d55.0ff3b2",

"type": "function",

"z": "6e79dbc7.7e3c14",

"name": "",

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

"outputs": 1,

"noerr": 0,

"x": 590,

"y": 300,

"wires": [

[

"f2f2649a.0d0d98",

"77f92f28.edd7f8"

]

]

},

{

"id": "77f92f28.edd7f8",

"type": "ui\_text",

"z": "6e79dbc7.7e3c14",

"group": "e06c378a.d686d",

"order": 3,

"width": 8,

"height": "8",

"name": "",

"label": "",

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

"layout": "col-center",

"x": 750,

"y": 360,

"wires": []

},

{

"id": "bd43da0c.bb3f28",

"type": "debug",

"z": "6e79dbc7.7e3c14",

"name": "",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "payload",

"targetType": "msg",

"x": 690,

"y": 80,

"wires": []

},

{

"id": "cc13951b.ef59b",

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

"z": "6e79dbc7.7e3c14",

"name": "",

"workspaceid": "04767092-8b24-4433-b9a9-127fb6f52a8d",

"multiuser": false,

"context": true,

"empty-payload": false,

"service-endpoint": "https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/b583a409-8d48-4d62-8f48-380efbc5c51e",

"timeout": "",

"optout-learning": false,

"x": 400,

"y": 180,

"wires": [

[

"3e2c8d55.0ff3b2",

"bd43da0c.bb3f28"

]

]

},

{

"id": "6aed6986.c684f",

"type": "ui\_text",

"z": "6e79dbc7.7e3c14",

"group": "e06c378a.d686d",

"order": 1,

"width": 7,

"height": 1,

"name": "",

"label": "You: ",

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

"layout": "row-left",

"x": 360,

"y": 360,

"wires": []

}

]

**Watson Cloud Function Action Code**

const assert = require('assert');

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

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,

22

'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);

});

23