

**PROJECT REPORT**  
**ON**  
**INTELLIGENT CUSTOMER HELP DESK WITH**  
**SMART DOCUMENT UNDERSTANDING**  
**(ARTIFICIAL INTELLIGENCE)**

**BY**  
**NEETI MEHRA**  
**([neetimehra23@gmail.com](mailto:neetimehra23@gmail.com))**

<b>1</b>	<b>INTRODUCTION</b>
	1.1 Overview
	1.2 Purpose
<b>2</b>	<b>LITERATURE SURVEY</b>
	2.1 Existing problem
	2.2 Proposed solution
<b>3</b>	<b>THEORITICAL ANALYSIS</b>
	3.1 Block diagram
	3.2 Hardware / Software designing
<b>4</b>	<b>EXPERIMENTAL INVESTIGATIONS</b>
<b>5</b>	<b>FLOWCHART</b>
<b>6</b>	<b>RESULT</b>
<b>7</b>	<b>ADVANTAGES &amp; DISADVANTAGES</b>
<b>8</b>	<b>APPLICATIONS</b>
<b>9</b>	<b>CONCLUSION</b>
<b>10</b>	<b>FUTURE SCOPE</b>
<b>11</b>	<b>BIBILOGRAPHY</b>
	<b>APPENDIX</b>
	A. Source code

# **1. INTRODUCTION**

## **1.1 OVERVIEW**

We will be able to write an application that leverages multiple Watson AI Services (Discovery , Assistant, Cloud function and Node Red). By the end of the project, we'll learn best practices of combining Watson services, and how they can build interactive information retrieval systems with Discovery + Assistant.

## **1.2 PURPOSE**

### **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**

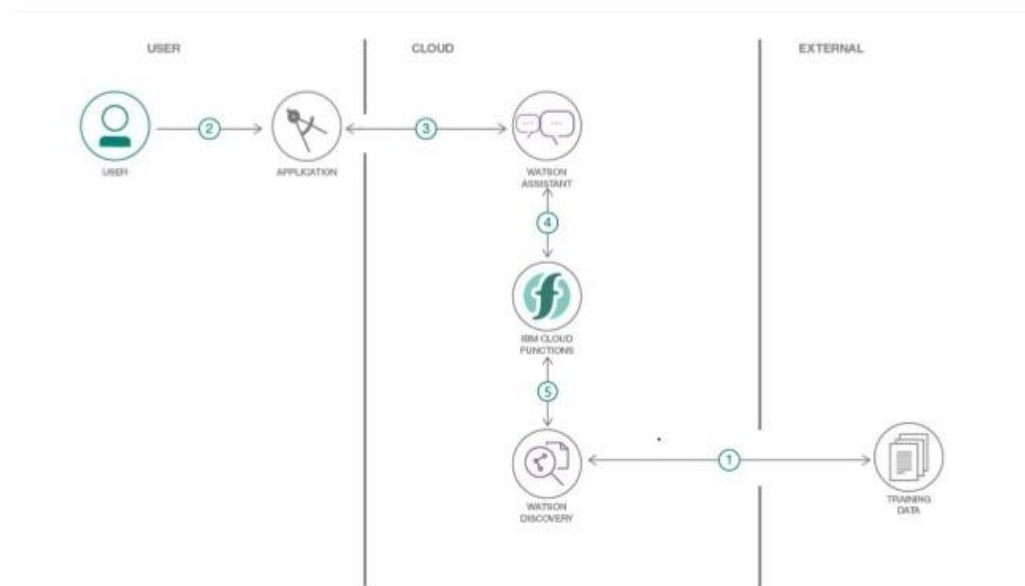
The typical customer care chatbot can answer simple questions, such as store locations and hours, directions, and maybe even making appointments. When a question falls outside of the scope of the pre-determined question set, the option is typically to tell the customer the question isn't valid or offer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

## **2.2 PROPOSED SOLUTION**

To solve the above mentioned problem we will introduce a Watson Discovery feature that is Smart Document Understanding. 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.

### 3. THEORITICAL ANALYSIS

#### 3.1 BLOCK DIAGRAM



1. The document is annotated using Smart Document Understanding feature of Watson Discovery.
2. The user interacts with the backend server via the app UI. The frontend 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 query about product, 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.

#### 3.2 HARDWARE/SOFTWARE DESIGNING

1. Create IBM Cloud services
2. Configure Watson Discovery
3. Create IBM Cloud Functions action
4. Configure Watson Assistant
5. Create flow and configure node
6. Deploy and run Node Red app.

## 4. EXPERIMENTAL INVESTIGATIONS

### 4.1 CREATE IBM CLOUD SERVICES

Create the following services:

- A. Watson Discovery
- B. Watson Assistant
- C. Node Red

### 4.2 CONFIGURE WATSON DISCOVERY

#### 4.2.1 Import the document

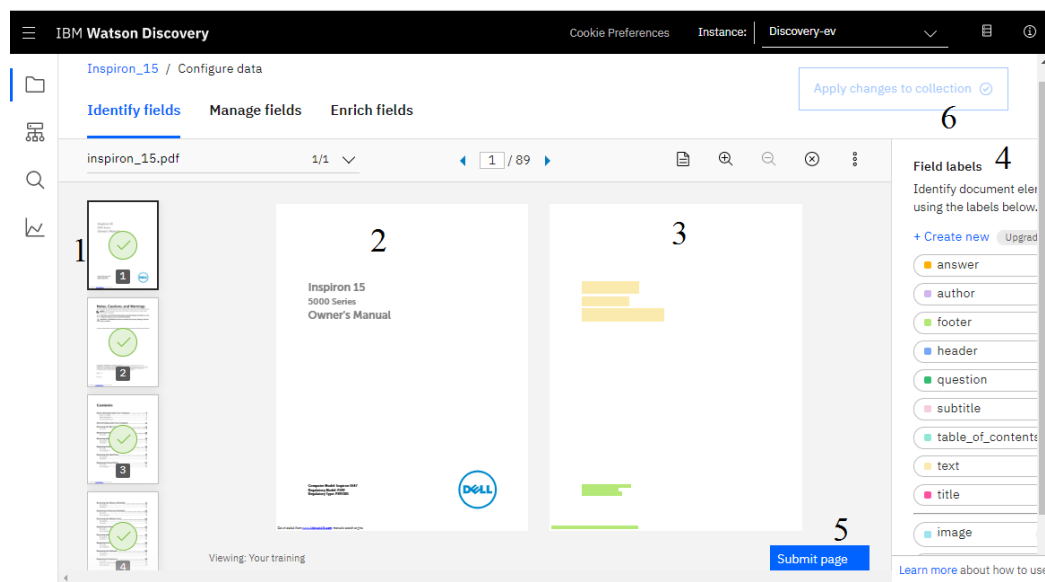
Launch the Watson Discovery tool and create a new data collection by selecting the Upload your own data option. Give the data collection a unique name. When prompted, select and upload the inspiron\_15.pdf file located in the data directory of your local repo.

This inspiron\_15 file contains configuration of different parts in Dell inspiron\_15 laptop model.

#### 4.2.2 Annotate with SDU

Now let's apply SDU to our document to see if we can generate some better query responses. From the Discovery collection panel, click the Configure data button (located in the top right corner) to start the SDU process.

Here is the layout of the Identify fields tab of the SDU annotation panel:



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.

[1] is the list of pages in the manual. As each is processed, a green check mark will appear on the page.

[2] is the current page being annotated.

[3] is where you select text and assign it a label.

[4] is the list of labels you can assign to the page text.

Click [5] to submit the page to Discovery.

Click [6] when you have completed the annotation process.

As you go through the annotations one page at a time, Discovery is learning and should start automatically updating the upcoming pages. Once you get to a page that is already correctly annotated, you can stop, or simply click Submit [5] to acknowledge it is correct. The more pages you annotate, the better the model will be trained.

For this specific owner's manual, at a minimum, it is suggested to mark the following:

The main title page as title

All headers and sub-headers as subtitle.

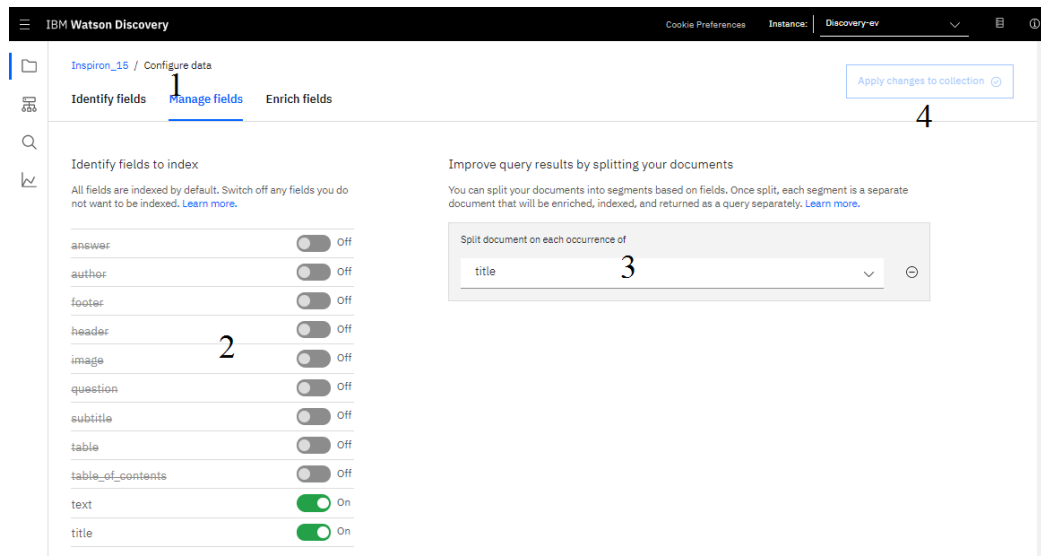
All page numbers as footers.

All warranty and licensing information (located in the last few pages) as a footer.

All other text should be marked as text.

Once you click the Apply changes to collection button [6], you will be asked to reload the document. Choose the same owner's manual .pdf document as before.

Next, click on the Manage fields [1] tab.



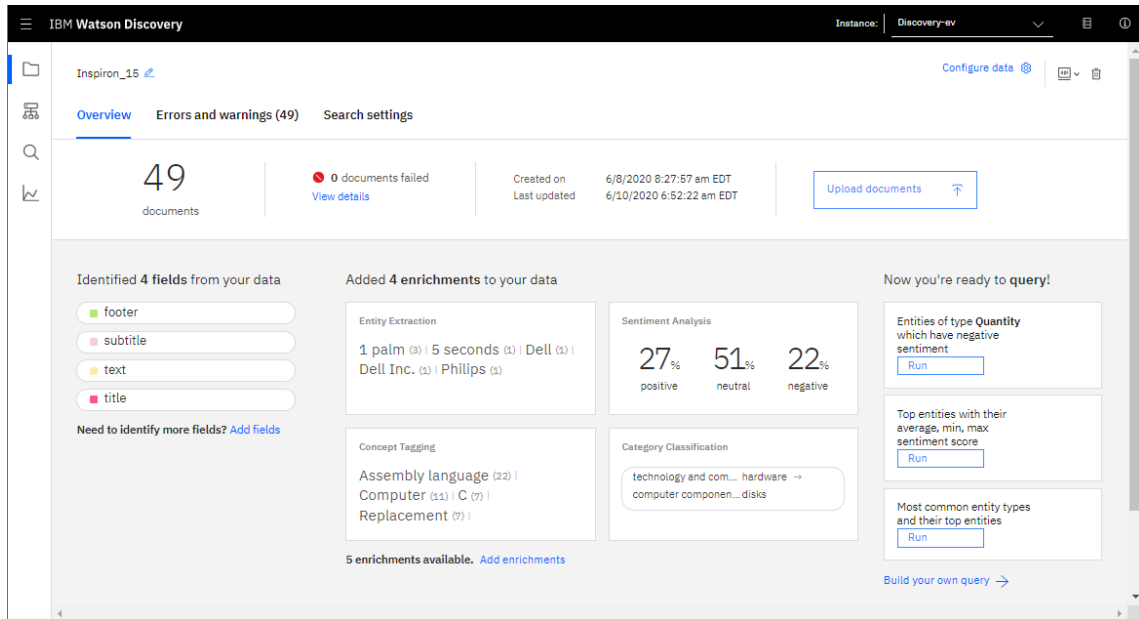
[2] Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except text and title.

[3] is telling Discovery to split the document apart, based on title.

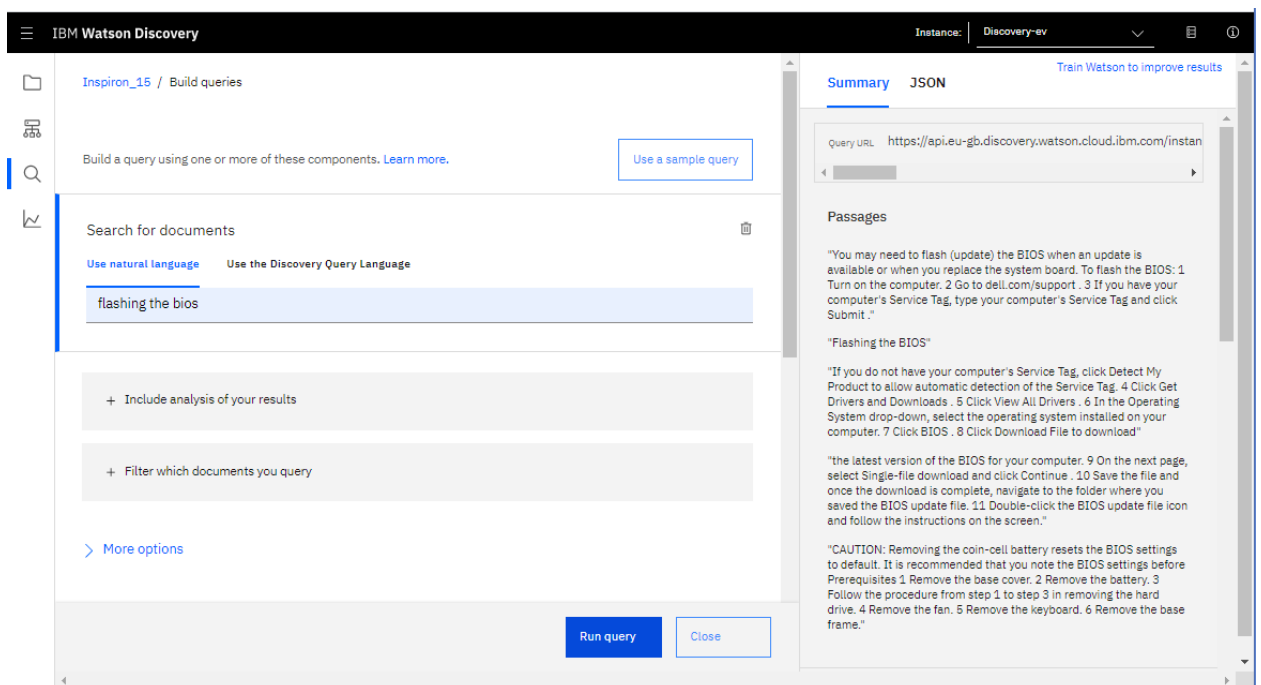
Click [4] to submit your changes.

Once again, you will be asked to reload the document.

Now, as a result of splitting the document apart, your collection will look different:



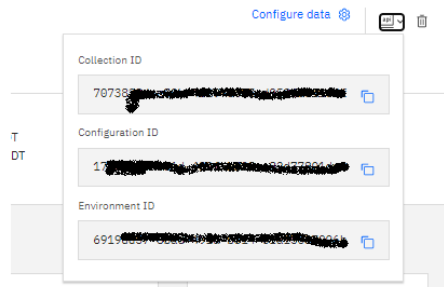
Go to Build your own query panel and see how much better the results are.



Store credentials for future use.

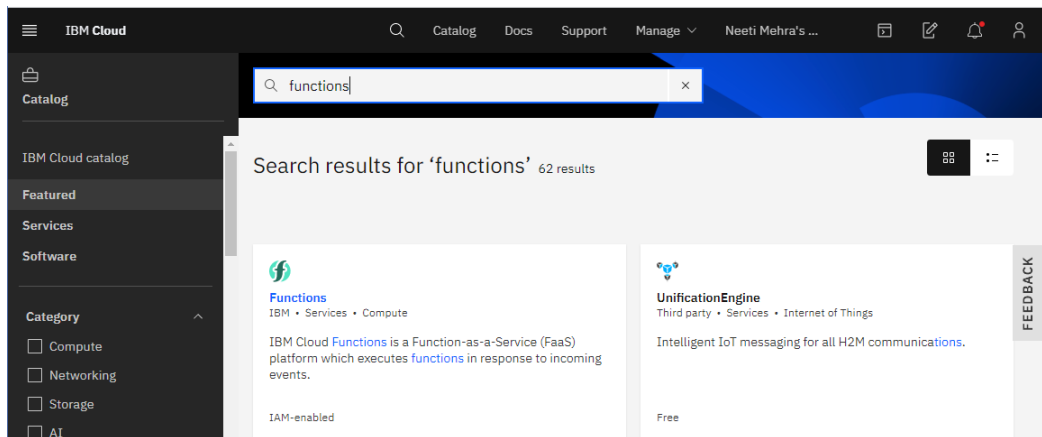
In upcoming steps, you will need to provide the credentials to access your Discovery collection. The values can be found in the following locations.

The Collection ID, Environment ID and Configuration ID values can be found by clicking the dropdown button located at the top right side of your collection panel:



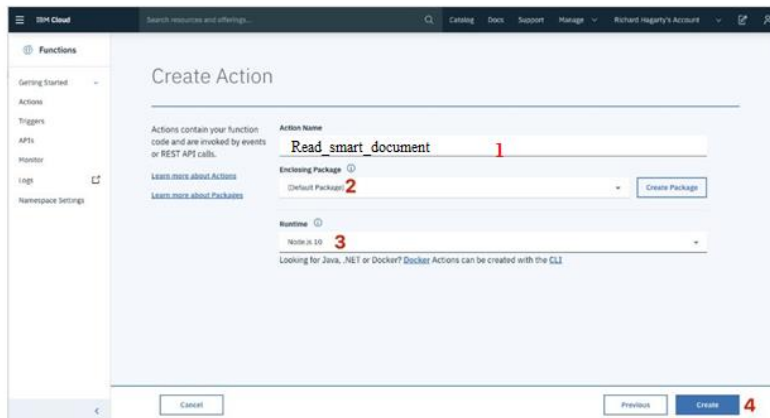
### 4.3 CREATE IBM CLOUD FUNCTION ACTIONS

Now let's create the web action that will make queries for our Discovery collection. Go to IBM Cloud dashboard and in catalog section select Function to create the action.



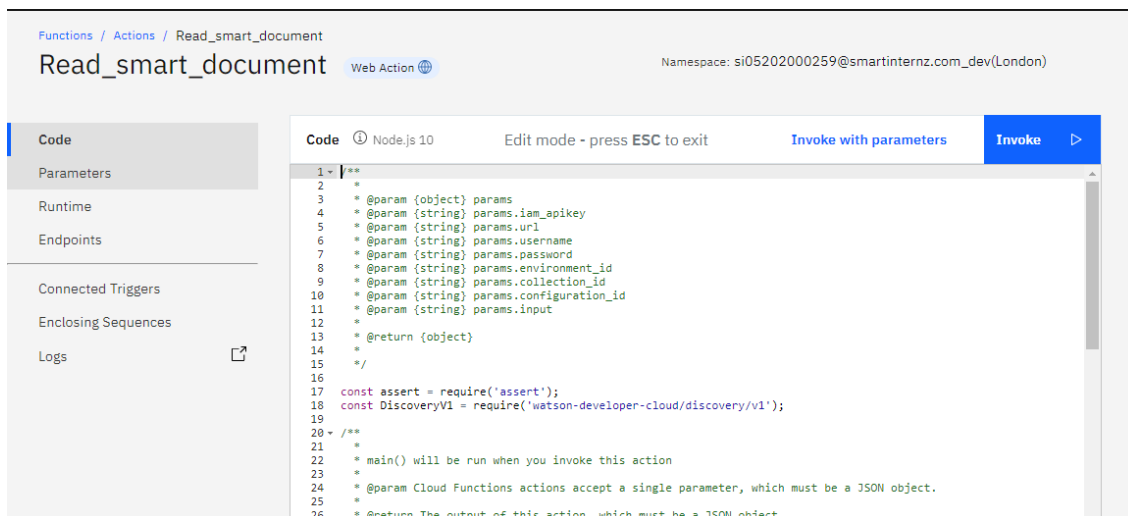
From the Functions main panel, click on the Actions tab. Then click on Create. From the Create panel, select the Create Action option.

On the Create Action panel, provide a unique Action Name [1], keep the default package [2], and select the Node.js 10 [3] runtime. Click the Create button [4] to create the action.





Once your action is created, click on the Code tab:

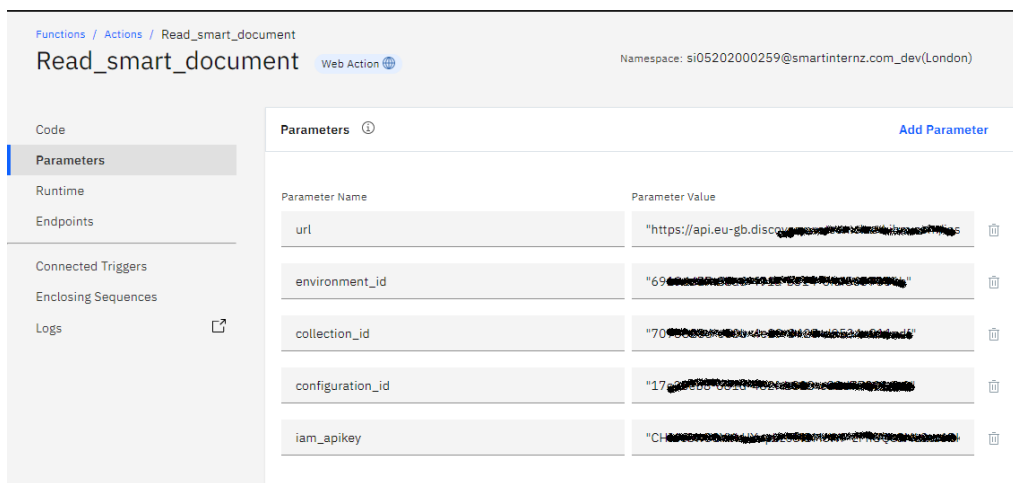


```
1 /**
2  *
3  * @param {object} params
4  * @param {string} params.iam_apikey
5  * @param {string} params.url
6  * @param {string} params.username
7  * @param {string} params.password
8  * @param {string} params.environment_id
9  * @param {string} params.collection_id
10 * @param {string} params.configuration_id
11 * @param {string} params.input
12 *
13 * @return {object}
14 */
15
16
17 const assert = require('assert');
18 const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');
19
20 /**
21 *
22 * main() will be run when you invoke this action
23 *
24 * @param Cloud Functions actions accept a single parameter, which must be a JSON object.
25 *
26 * @return The output of this action, which must be a JSON object.
```

In the code editor window, cut and paste in the respective code. The code is pretty straight-forward - it simply connects to the Discovery service, makes a query against the collection, then returns the response.

If you press the Invoke button, it will fail due to credentials not being defined yet. We'll do this next.

Select the Parameters tab



Parameter Name	Parameter Value
url	"https://api.eu-gb.discovery.watson.ibm.com/discovery/v1" [icon]
environment_id	"69 [redacted]" [icon]
collection_id	"70 [redacted]" [icon]
configuration_id	"17 [redacted]" [icon]
iam_apikey	"Cl [redacted]" [icon]

Add the following keys – url, environment\_id, collection\_id, configuration\_id and 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:

Functions / Actions / Read\_smart\_document

## Read\_smart\_document Web Action

Namespace: si05202000259@smartinternz.com\_dev(London)

Code

Parameters

Runtime

Endpoints

Connected Triggers

Enclosing Sequences

Logs

Code Node.js 10 Invoke with parameters Invoke

```

1  /**
2   *
3   * @param {object} params
4   * @param {string} params.iam_apikey
5   * @param {string} params.url
6   * @param {string} params.username
7   * @param {string} params.password
8   * @param {string} params.environment_id
9   * @param {string} params.collection_id
10  * @param {string} params.configuration_id
11  * @param {string} params.input
12  *
13  * @return {object}
14  *
15  */
16
17 const assert = require('assert');
18 const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');
19
20 /**
21 *
22 * main() will be run when you invoke this action
23 *
24 * @param Cloud Functions actions accept a single parameter, which must be
25 *
26 * @return The output of this action, which must be a JSON object.
27 *
28 */
29 function main(params) {
30

```

Activations Collapse Clear

Read\_smart\_document 1049 ms 6/11/2020, 13:43:57

Activation ID: a793ff36356e467893ff36356e67836

Results:

```

{
  "matching_results": 49,
  "passages": [
    {
      "results": [
        {
          "enriched_text": {
            "categories": [
              {
                "label": "/technology and computing/hardware/computer peripherals",
                "score": 0.99996
              }
            ]
          },
          "label": "/technology and computing/hardware/computer",
          "score": 0.99996
        },
        {
          "label": "/technology and computing/hardware/computer peripherals/computer monitors",
          "score": 0.999917
        }
      ]
    }
  ]
}

```

Next, go to the Endpoints panel [1]

Read\_smart\_document Web Action

Namespace: si05202000259@smartinternz.com\_dev(London)

Code

Parameters

Runtime

**Endpoints 1**

Connected Triggers

Enclosing Sequences

Logs

**Web Action**

☒ **Enable as Web Action** Allow your Cloud Functions actions to handle HTTP events. Web Actions allow to control the response data and type by using a set of URL extensions, such as .json or .html. Learn more about [Web Actions](#). Note: The Web Action URL below requires to return a dict object that contains a body property.

☐ **Raw HTTP handling** When enabled your Action receives requests in plain text instead of a JSON body

HTTP Method	Auth	URL
ANY	Public	<a href="https://eu-gb.functions.cloud.ibm.com/api/v1/web/si05202000259%40smartinternz.com_dev/default/Read_smart_document">https://eu-gb.functions.cloud.ibm.com/api/v1/web/si05202000259%40smartinternz.com_dev/default/Read_smart_document</a>

Click the checkbox for Enable as Web Action [2]. This will generate a public endpoint URL [3].

Take note of the URL value [3], as this will be needed by Watson Assistant in a future step.

## 4.4 CONFIGURE WATSON ASSISTANT

Launch the Watson Assistant tool and create a new dialog skill. This dialog skill contains all of the nodes needed to have a typical call center conversation with a user.

### 4.4.1 Add new intent

The default customer care dialog does not have a way to deal with any questions involving outside resources, so we will need to add this.

Create a new intent that can detect when the user is asking about operating the Laptop.

From the Customer Help Desk Skill panel, select the Intents tab.

Click the Create intent button.

Name the intent #Product\_Information, and at a minimum, enter the following example questions to be associated with it.

#Product\_information

Intent name

Name your intent to match a customer's question or goal

#Product\_information

Description (optional)

Add a description to this intent

User example

Add unique examples of what the user might say. (Pro tip: Add at least 5 unique examples to help Watson understand)

Type a user example here, e.g. I want to pay my credit card bill

Add example

Show recommendations

User examples (5) ↑

base frame

palm rest

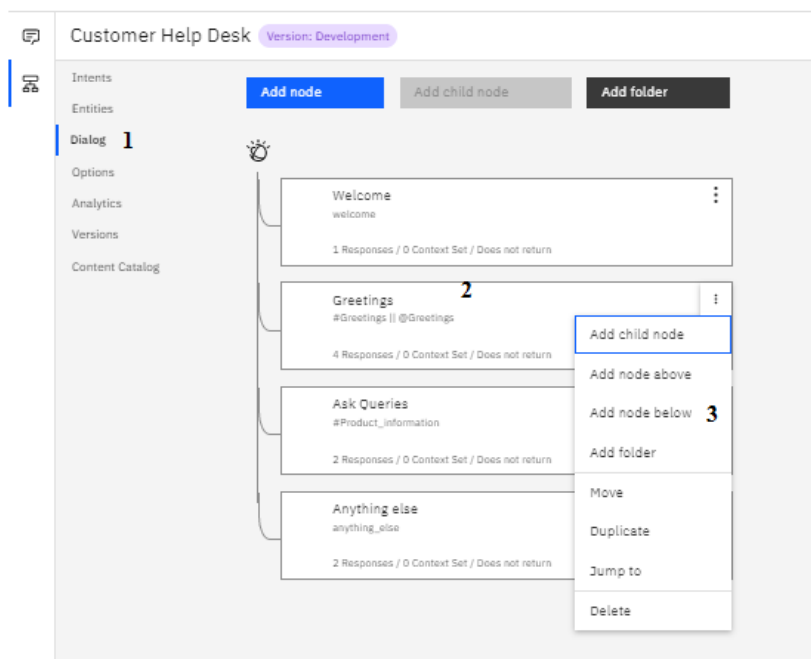
power adapter port

system board

wireless card

#### 4.4.2 Create new dialog node

Now we need to add a node to handle our intent. Click on the Dialog [1] tab, then click on the drop down menu for the Greetings node [2], and select the Add node below [3] option.



Name the node "Ask Queries" [1] and assign it our new intent [2].

The screenshot shows the Watson Assistant interface for a project named 'Customer Help Desk'. On the left, a sidebar lists various components: Intents, Entities, Dialog, Options, Analytics, Versions, and Content Catalog. The 'Dialog' tab is selected, showing a flowchart of dialog nodes. A node named 'Ask Queries' is highlighted with a blue border and labeled with a '1'. To the right, the configuration panel for the 'Ask Queries' node is displayed. It has a title 'Ask Queries' and a 'Customize' link. Below the title, it says 'Node name will be shown to customers for disambiguation to use something descriptive.' and a 'Settings' link. Under the heading 'If assistant recognizes', there is a list of intents. The first intent is '#Product\_information', which is selected and labeled with a '2'. Below this, under the heading 'Then callout to my webhook', there is a 'Parameters' section. It contains a table with two columns: 'Key' and 'Value'. The first row has 'input' as the key and '<input\_text>' as the value. There is an 'Add parameter +' link at the bottom.

This means that if Watson Assistant recognizes a user input such as "how to flash the bios?", it will direct the conversation to this node.

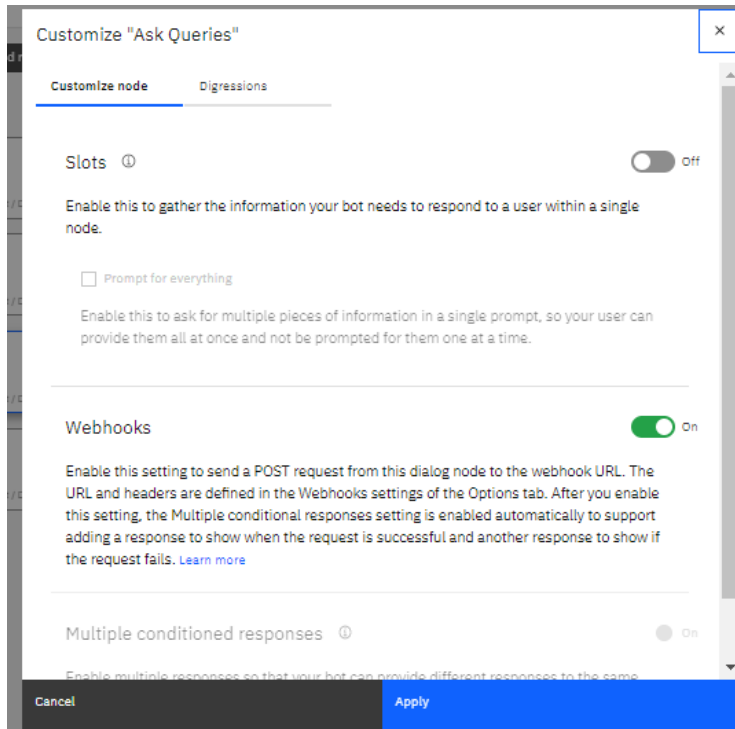
#### 4.4.3 ENABLE WEBHOOK FROM ASSISTANT

Set up access to our WebHook for the IBM Cloud Functions action you created in Step #4. Select the Options tab [1]:

The screenshot shows the 'Webhooks' configuration page in the Watson Assistant interface. The sidebar on the left is the same as in the previous screenshot, but the 'Options' tab is selected, and the 'Webhooks' sub-tab is highlighted with a blue border and labeled with a '1'. The main content area is titled 'Webhooks' and contains a description: 'A webhook is a mechanism that allows you to call out to an external program based on events in your dialog.' Below this, there is a 'Webhook setup' section. It says 'Specify the request URL for an external API you want to be able to invoke from dialog nodes. Watson will call this URL when configured to do so from a dialog node.' and a 'Learn more' link. Under the heading 'URL', there is a text input field containing the URL 'https://eu-gb.functions.cloud.ibm.com/api/v1/web/si05202000259%40smi' and labeled with a '2'. Below the URL field, there is a 'Headers' section. It says 'Add HTTP headers for authorization or any other parameters required for invoking the webhook.' Below this, there is a table with two columns: 'Header name' and 'Header value'. At the bottom of the table, there are two links: 'Add header +' and 'Add authorization +'. Below the table, there is a 'Next step' section. It says 'To trigger this webhook from an individual dialog node, enable webhooks from the Customize page of the node.' and a 'Go to dialog' link.

Enter the public URL endpoint for your action [2].

Return to the Dialog tab, and click on the Ask Queries node. From the details panel for the node, click on Customize, and enable Webhooks for this node:



Click Apply.

The dialog node should have a Return variable [1] set automatically to \$webhook\_result\_1. This is the variable name you can use to access the result from the Discovery service query.

If assistant recognizes

#Product\_information

Then callout to my webhook [Learn more](#)

Key	Value
2 input	"<?input.text?>"

[Add parameter +](#)

Return variable

1 \$webhook\_result\_1

You will also need to pass in the users question via the parameter input [2]. The key needs to be set to the value: "<?input.text?>"

If you fail to do this, Discovery will return results based on a blank query. Optionally, you can add these responses to aid in debugging:

#### Return variable

\$webhook\_result\_1

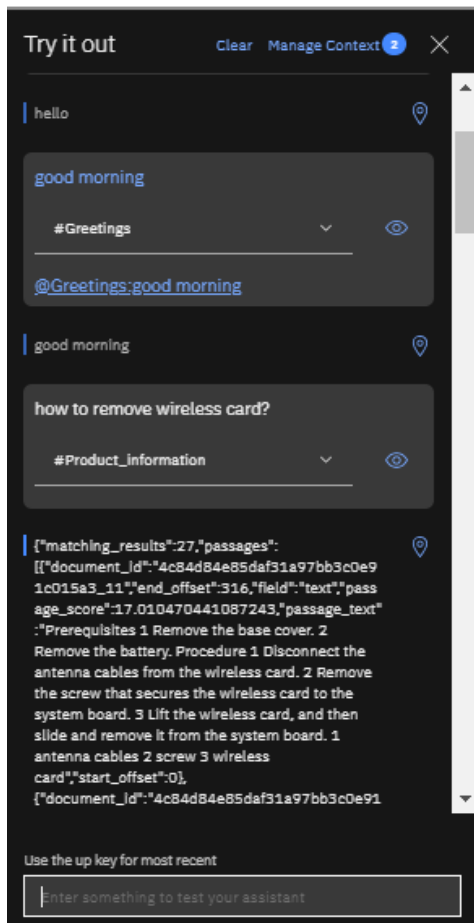
#### Assistant responds

	If assistant recognizes	Respond with		
1	\$webhook_result_1	\$webhook_result_1	⚙️	—
2	anything_else	ERROR	⚙️	—

#### 4.4.4 Test in Assistant Tooling

From the Dialog panel, click the Try it button located at the top right side of the panel.

Enter some user input:



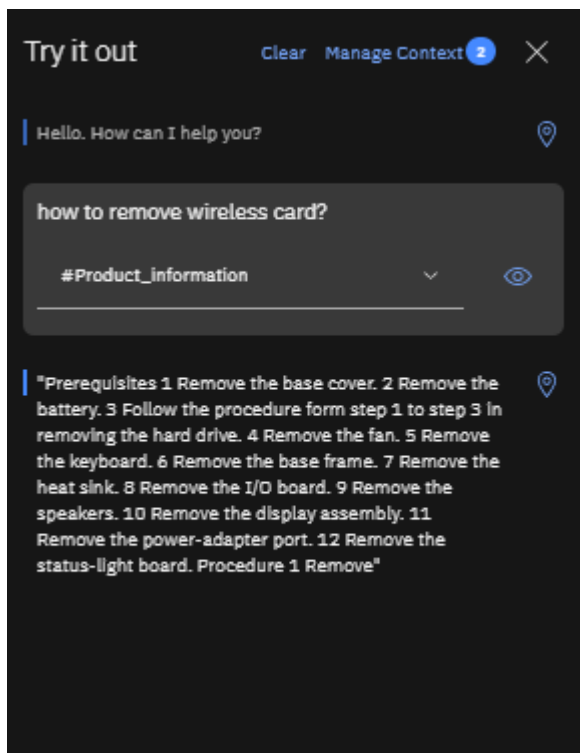
Note that the input "how to remove wireless card?" has triggered our Ask Queries dialog node, which is indicated by the #Product\_Information response.

And because we specified that \$webhook\_result\_1.passages be the response, that value is displayed also.

For better and accurate results we can change the “Respond with” tab as "<?\$webhook\_result\_1.passages[0].passage\_text?>" and try results again.

#### Assistant responds

	If assistant recognizes	Respond with		
1	\$webhook_result_1	"<?\$webhook_result_1.passa	⚙️	—
2	anything_else	ERROR	⚙️	—



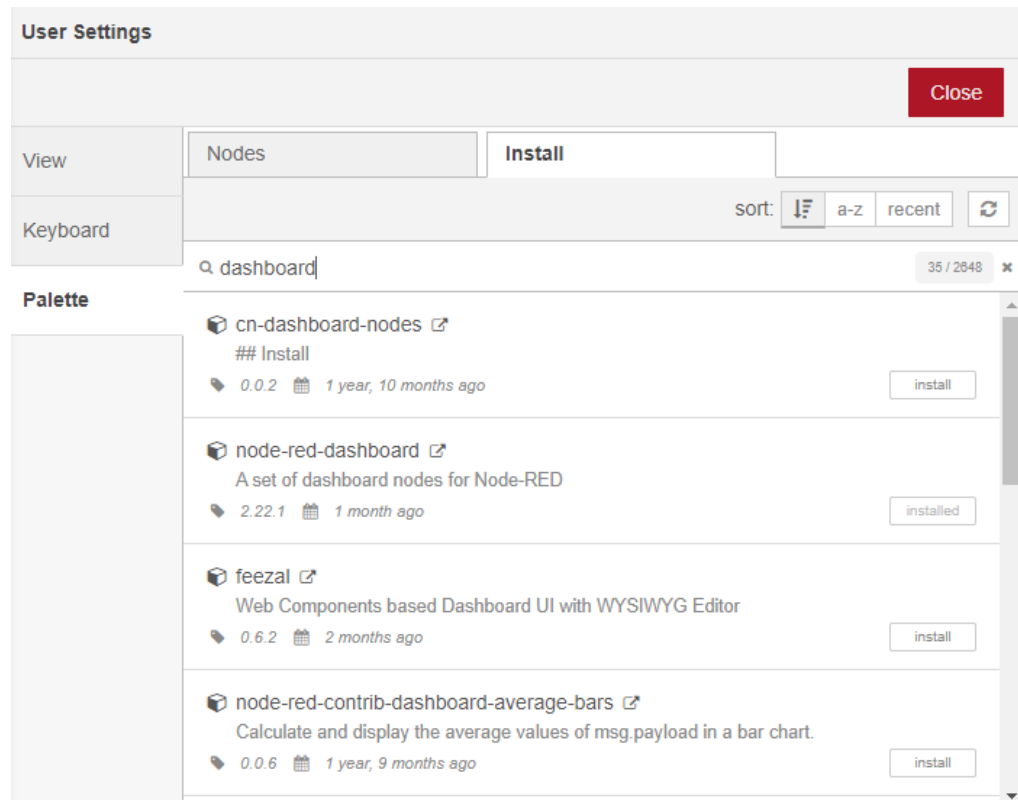
## 4.5 CREATE FLOW AND CONFIGURE NODE

### 4.5.1 Integration of watson assistant in Node-RED

- Double-click on the Watson assistant node
- Give workspace id of your Watson assistant service







- Click on install
- Search for “node-red-dashboard” and click on install and again click on install on the prompt
- The following message indicates dashboard nodes are installed, close the manage palette
- Search for “Form” node and drag on to the flow
- Double click on the “form” node to configure
- Click on the edit button to add the “Group” name and “Tab” name
- Click on the edit button to add tab name to web application
- Give sample tab name and click on add do the same thing for the group
- Give the label as “Enter your input”, Name as “text” and click on Done
- Drag a function node, double-click on it and enter the input parsing code as shown below

**Edit function node**

Delete Cancel Done

⚙️ **Properties** ⚙️ 📄 🖨️

🔑 Name  📄 ▼

🔑 Function 🔄

```
1 msg.payload = msg.payload.text;
2 return msg;
```

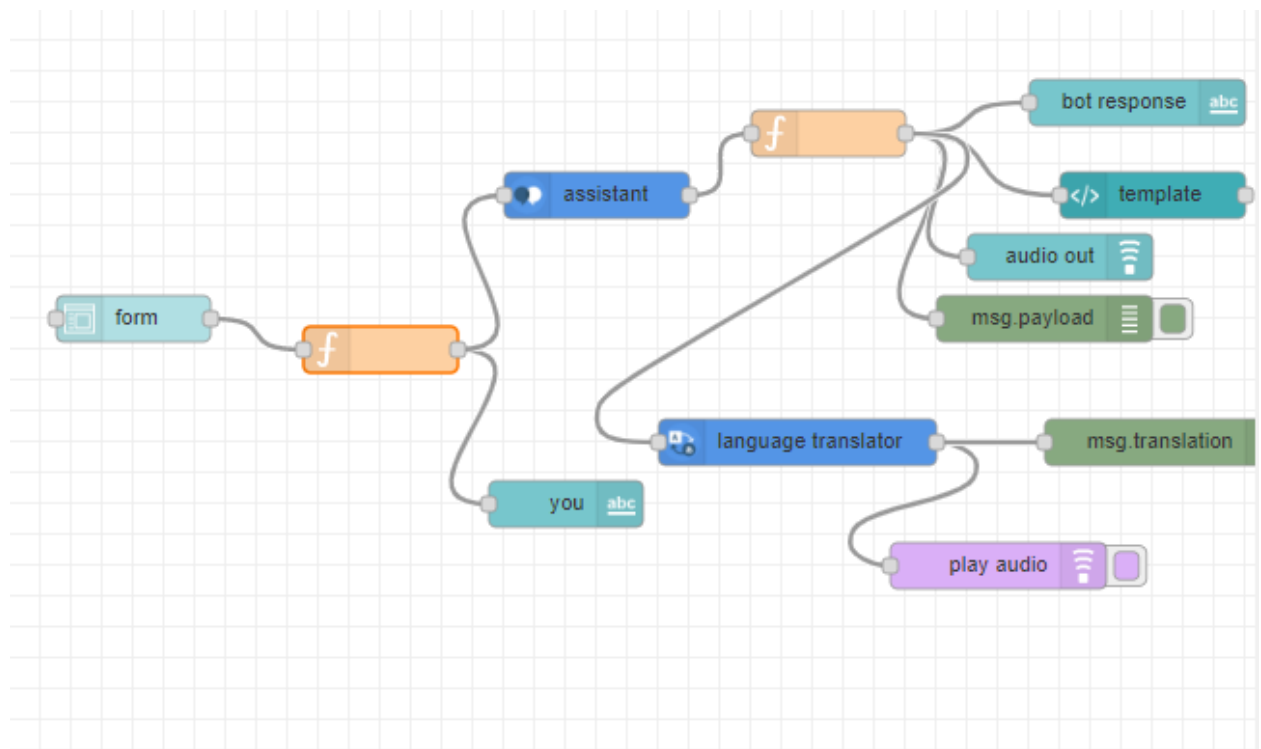
- Click on done
- Connect the form output to the input of the function node and output of the function to input of assistant node
- Search for “text” node from the “dashboard” section
- Drag two “text” nodes on to the flow
- Double click on the first text node, change the label as “You” and click on Done
- Double click on the second text node, change the label as “Bot” and click on Done
- Connect the output of “input parsing” function node to “ You” text node and output of “Parsing” function node to the input of “Bot” text node
- Click on Deploy

## 5. FLOWCHART

Firstly, go to manage palette and install dashboard.

Now create the flow with the help of following node:

- UI\_Form
- Function
- Assistant
- Debug
- UI\_text
- Template
- Audio\_out
- Language translator
- Play audio



## 6. RESULTS

Finally our Node-RED dash board integrates all the components and displayed in the Dashboard UI by typing URL - <https://node-red-qgwak.eu-gb.mybluemix.net/ui> in browser.

## Customer Help Desk

### DellBot

Enter input \*

hi

SUBMIT

RESET

you

hi

bot response

**Hello! I am a DellBot. I can provide you information regarding Inspiron\_15 Laptop !!!**

## Customer Help Desk

### DellBot

Enter input \*

how to flash the bios?

SUBMIT

RESET

you

**how to flash the bios?**

bot response

**"You may need to flash (update) the BIOS when an update is available or when you replace the system board. To flash the BIOS: 1 Turn on the computer. 2 Go to dell.com/support . 3 If you have your computer's Service Tag, type your computer's Service Tag and click Submit ."**

## 7. PRONS AND CONS

### PROS:

- Companies can deploy chatbots to rectify simple and general human queries.
- Reduces man power
- Cost efficient
- No need to divert calls to customer agent and customer agent can look on other works.

### CONS:

- Some times chatbot can mislead customers
- Giving same answer for different sentiments.
- Sometimes cannot connect to customer sentiments and intentions.

## 8. APPLICATIONS

- It can deploy in popular social media applications like facebook messenger, slack, telegram.
- Chatbot can deploy any website to clarify basic doubts of viewers.

## 9. CONCLUSION

By doing the above procedure and all we successfully created Intelligent helpdesk smart chartbot using Watson assistant, Watson discovery, Node-RED and cloud-functions.

## 10. FUTURE SCOPE

We can include watson studio text to speech and speech to text services to access the chatbot handsfree. This is one of the future scope of this project.

## 11. BIBILOGRAPHY

### APPENDIX

#### SOURCE CODE

##### A. CLOUD FUNCTION – ACTION

```
/**
```

```
*
```

```
* @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);
  });
});
}

```

## B. NODE RED FLOW

```

[{"id":"b17fd556.2e6468","type":"tab","label":"Watson
Assistant","disabled":false,"info":"","{ "id":"d38e8f79.cf6d","type":"ui_form","z":"b17fd556.2e646
8","name":"","label":"","group":"37effca.047b004","order":17,"width":0,"height":0,"options":[{"la
bel":"Enter
input","value":"text","type":"text","required":true,"rows":null}],"formValue":{"text":"","payload"
":"","submit":"submit","cancel":"reset","topic":"","x":80,"y":200,"wires":[["2141a288.337fae"]]},{"
id":"2141a288.337fae","type":"function","z":"b17fd556.2e6468","name":"","func":"msg.payload =
msg.payload.text;\nreturn
msg;","outputs":1,"noerr":0,"x":240,"y":220,"wires":[["88c1c2b9.c6bd","79738be3.f8c6a4"]]},{"id
":"6e9114ac.a09cfc","type":"function","z":"b17fd556.2e6468","name":"","func":"if(msg.payload.o
utput.error)\n{\n  msg.payload=\"please rephrase\";\n  return msg;\n}\nmsg.payload =
msg.payload.output.text[0];\nreturn
msg;","outputs":1,"noerr":0,"x":530,"y":80,"wires":[["e2f4108e.8f6b8","713eca3b.0525e4","f8a7a
555.ae3668","763da3c9.f00c0c","ccaafd8.00791"]]},{"id":"713eca3b.0525e4","type":"debug","z":
"b17fd556.2e6468","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"compl
ete":"payload","targetType":"msg","x":670,"y":200,"wires":[]},{"id":"e2f4108e.8f6b8","type":"ui_
text","z":"b17fd556.2e6468","group":"37effca.047b004","order":19,"width":6,"height":4,"name"
":"","label":"bot response","format":"{{ msg.payload }}","layout":"col-
center","x":730,"y":60,"wires":[]},{"id":"88c1c2b9.c6bd","type":"ui_text","z":"b17fd556.2e6468",
"group":"37effca.047b004","order":18,"width":6,"height":2,"name":"","label":"you","format":"
{{ msg.payload }}","layout":"col-
center","x":360,"y":320,"wires":[]},{"id":"79738be3.f8c6a4","type":"watson-conversation-
v1","z":"b17fd556.2e6468","name":"","workspaceid":"66568b8e-db65-42da-8d6e-

```

```
27232d815f59","multiuser":false,"context":true,"empty-payload":false,"service-
endpoint":"","timeout":"","optout-
learning":false,"x":380,"y":120,"wires":[["6e9114ac.a09cfc"]]],{"id":"f8a7a555.ae3668","type":"ui
_audio","z":"b17fd556.2e6468","name":"","group":"37effca.047b004","voice":"en-
US","always":"","x":680,"y":160,"wires":[]},{ "id":"ccaafd8.00791","type":"watson-
translator","z":"b17fd556.2e6468","name":"","action":"translate","basemodel":"ar-
en","domain":"general","srclang":"en","destlang":"es","password":"","apikey":"","custom":"","do
mainhidden":"general","srclanghidden":"en","destlanghidden":"es","basemodelhidden":"ar-
en","customhidden":"","filetype":"forcedglossary","trainid":"","lgparams2":true,"service-
endpoint":"https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/79b7fcde-6287-464c-9842-
1e9a751d423e","x":510,"y":280,"wires":[["770f5d98.5d4e64","7de5d2c6.88a35c"]]],{"id":"770f5
d98.5d4e64","type":"play
audio","z":"b17fd556.2e6468","name":"","voice":"0","x":640,"y":360,"wires":[]},{ "id":"7de5d2c6.
88a35c","type":"debug","z":"b17fd556.2e6468","name":"","active":true,"tosidebar":true,"console":
false,"tostatus":false,"complete":"translation","targetType":"msg","x":750,"y":280,"wires":[]},{ "id
":"763da3c9.f00c0c","type":"ui_template","z":"b17fd556.2e6468","group":"37effca.047b004","na
me":"","order":20,"width":6,"height":1,"format":"","storeOutMessages":true,"fwdInMessages":tr
ue,"resendOnRefresh":true,"templateScope":"local","x":740,"y":120,"wires":[[]]},{ "id":"37effca.0
47b004","type":"ui_group","z":"","name":"chatbot","tab":"5daf2143.48c4","order":1,"disp":true,"
width":6,"collapse":false},{ "id":"5daf2143.48c4","type":"ui_tab","z":"","name":"Customer
Care","icon":"dashboard","disabled":false,"hidden":false}}
```

## REFERENCES

1. <https://developer.ibm.com/articles/introduction-watson-discovery/>
2. <https://medium.com/@rimaibrahim/node-red-watson-discovery-chatbot-telegram-ce616ddcd0d9>
3. <https://www.youtube.com/embed/Jpr3wVH3FVA>
4. <https://cloud.ibm.com/docs/openwhisk?topic=cloud-functions-getting-started>
5. <https://www.ibm.com/cloud/watson-assistant/>



