**PROJECT REPORT**

**ON**

**INTELLIGENT CUSTOMER HELP DESK WITH**

**SMART DOCUMENT UNDERSTANDING**

**BATCH**

**ARTIFICIAL INTELLIGENCE**

**BY**

**JATIN AGRAWAL**

**Jatinagrawal1516@gmail.com**

TABLE OF CONTENTS

1. INTRODUCTION
   1. Overview
   2. Purpose
2. LITERATURE SURVEY

2.1 Existing problem

2.2 Proposed solution

1. THEORITICAL ANALYSIS

3.1 Block diagram

3.2 Hardware / Software designing

1. EXPERIMENTAL INVESTIGATIONS
2. FLOWCHART
3. RESULT
4. ADVANTAGES & DISADVANTAGES
5. APPLICATIONS
6. CONCLUSION
7. FUTURE SCOPE `
8. REFERENCES
9. **INTRODUCTION**

In this project a chatbot is created which offers a complete and easy way to answer different sets of questions asked by the customers. With the help of Watson discovery channel it can also answer some typical questions about the operation of a device because we have feeds the owners manual to the watson discovery channel.

The benefits of this kind of chatbot is that it is superior than the typical chatbot which can answers simple questions like store location and hours. The chatbot is upgraded with the help of watson discovery collection which is build using smart document understanding.

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.

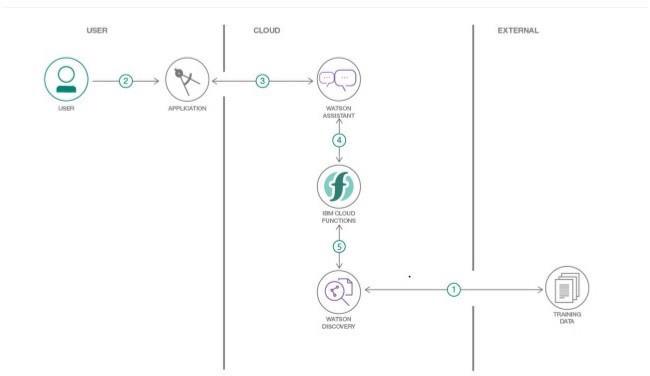
To take it a step further, the project shall use the Smart Document Understanding feature of Watson Discovery to train it on what text in the owners manual is important and what is not. This will improve the answers returned from the queries.

1.2 Purpose

* + To solve customer's queries as early as possible to save the time of the customer.
  + We will use the IBM cloud function that allows watson assistant to post queries to Watson discovery.
  + The goal is to set up a remote connection between the customer and the company.

1. **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.
7. **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

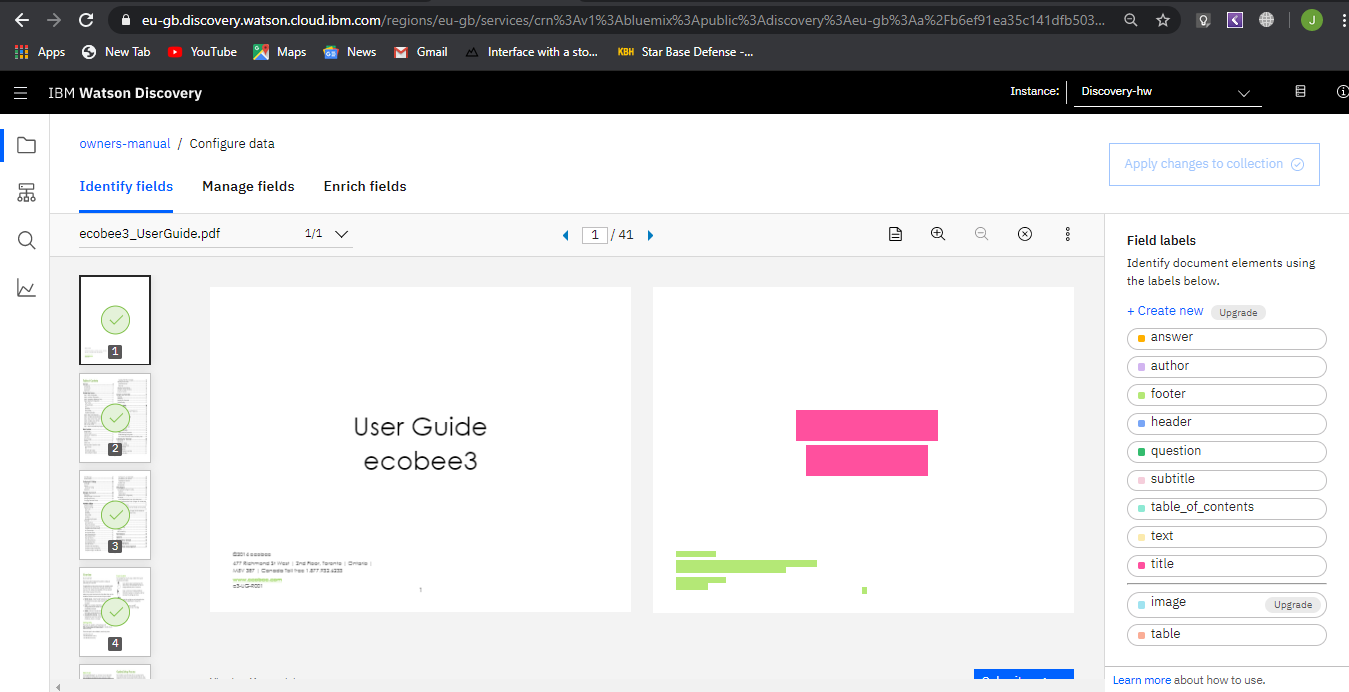
Launch the Watson Discovery tool and create a new data collection by selecting the Upload your own data option. Give the data collection a unique name. When prompted, select and upload the inspiron\_15.pdf file located in the data directory of your local repo.

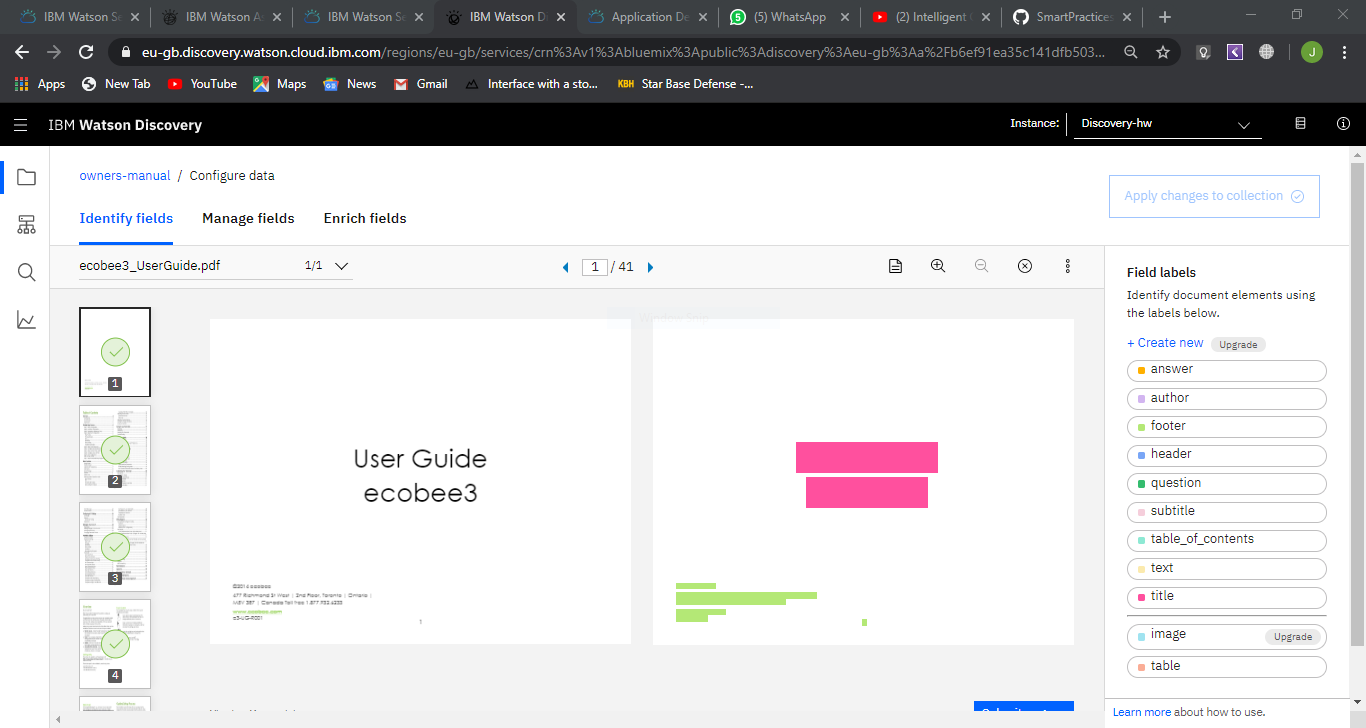
This Ecobee file contains configuration of different parts of Ecobee3 model.

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 annoted.
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 though 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

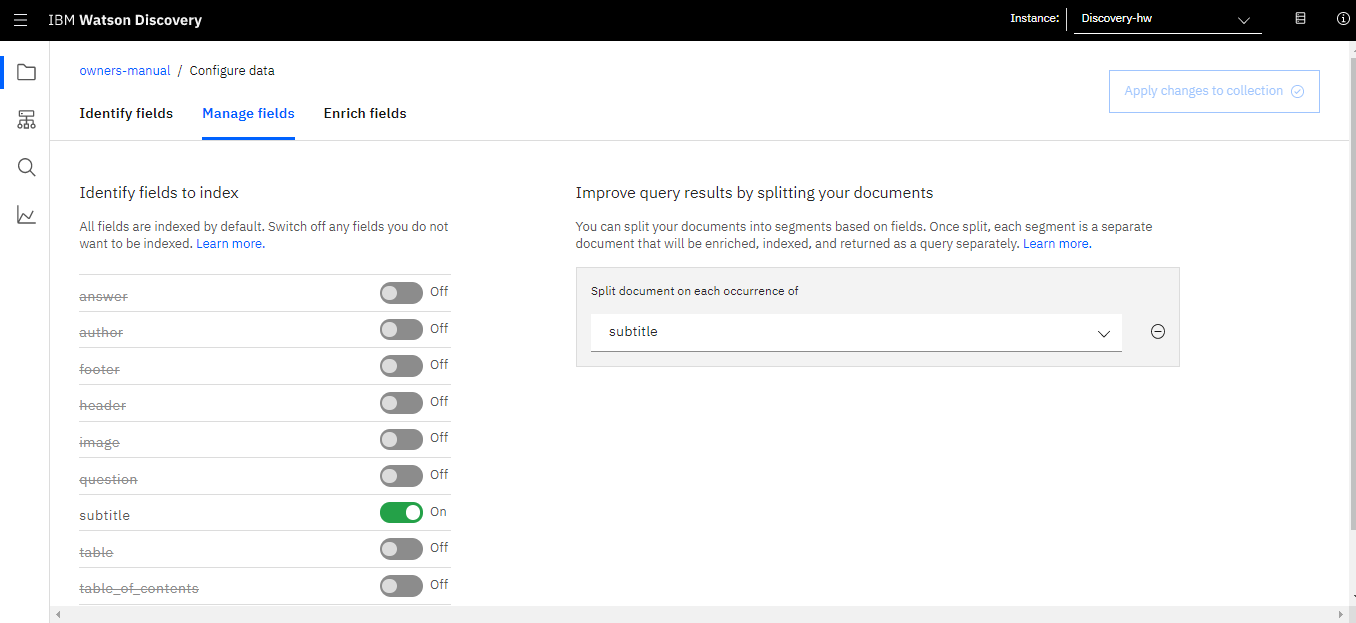
All headers and sub-headers as subtitle.

All page numbers as footers.

All warranty and licensing infomation (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.



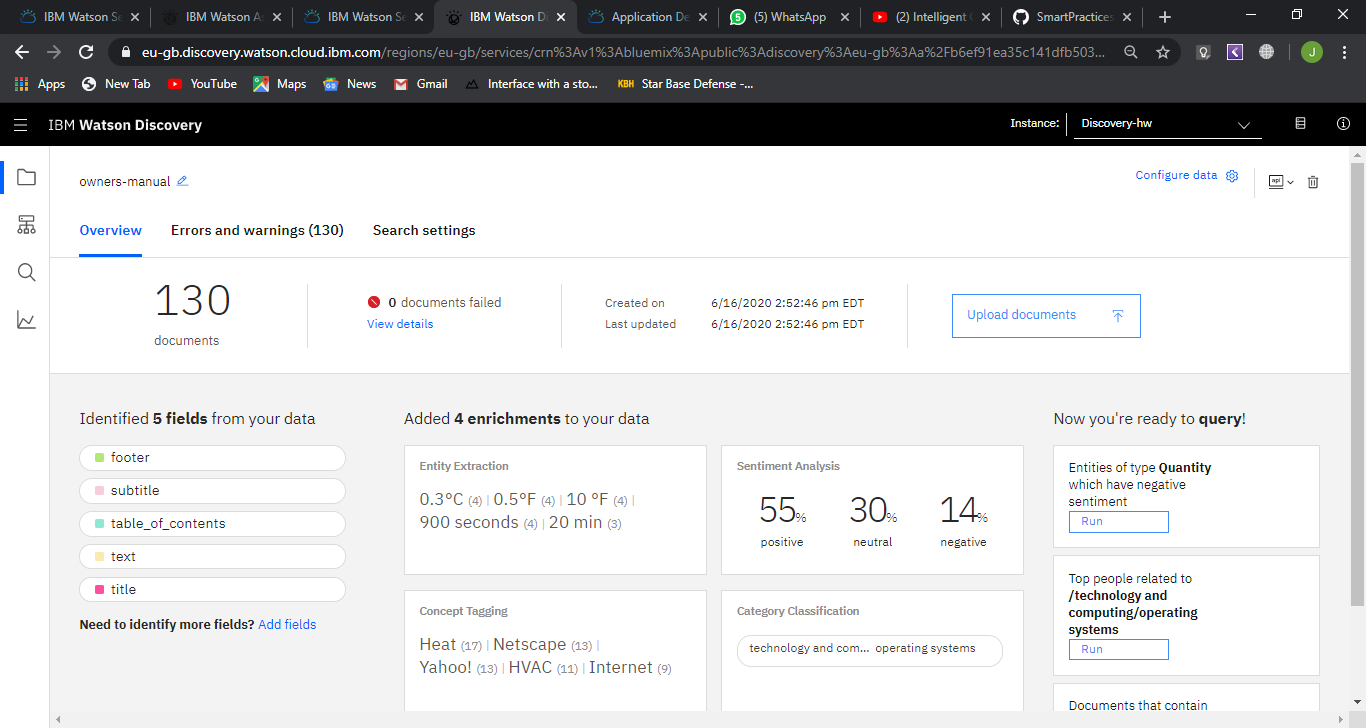
Next, click on the Manage fields tab.

1. Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except text and title.
2. is telling Discovery to split the document apart, based on title.

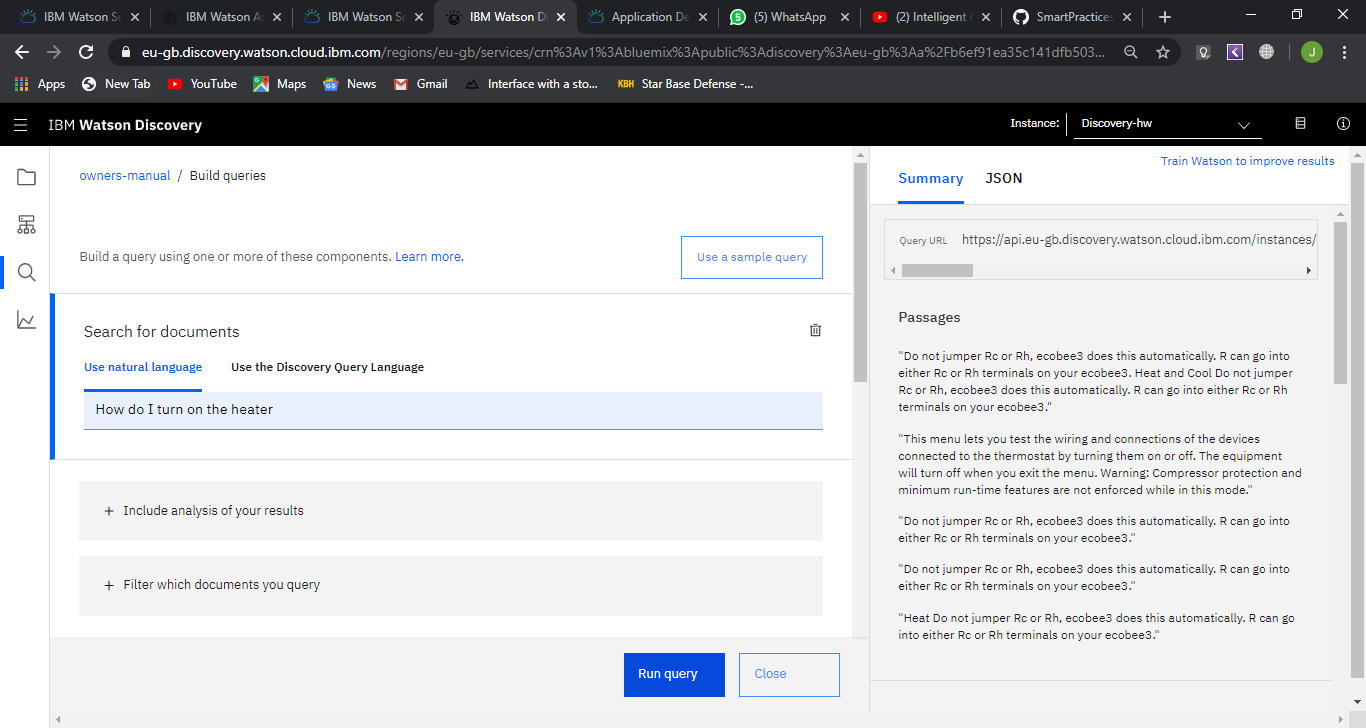
Click [4] to submit your changes.

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.

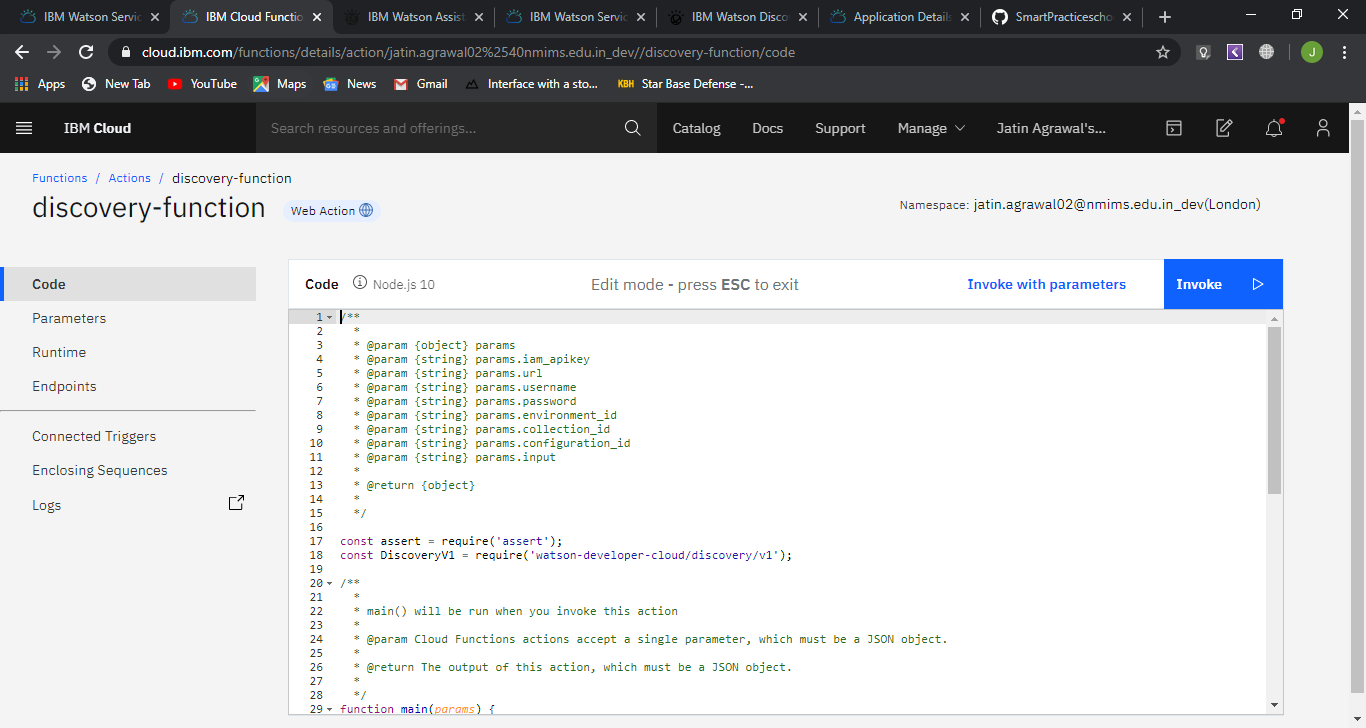


Store credentials for future use.

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:

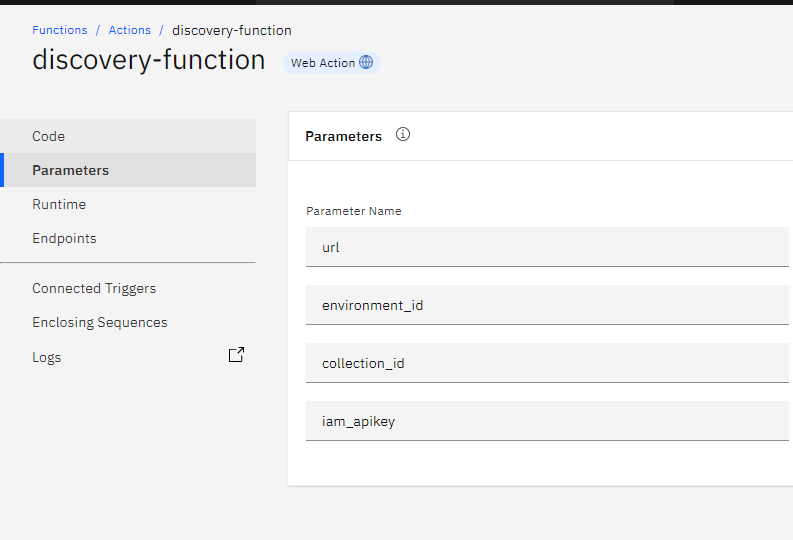
Once your action is created, click on the Code tab:



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

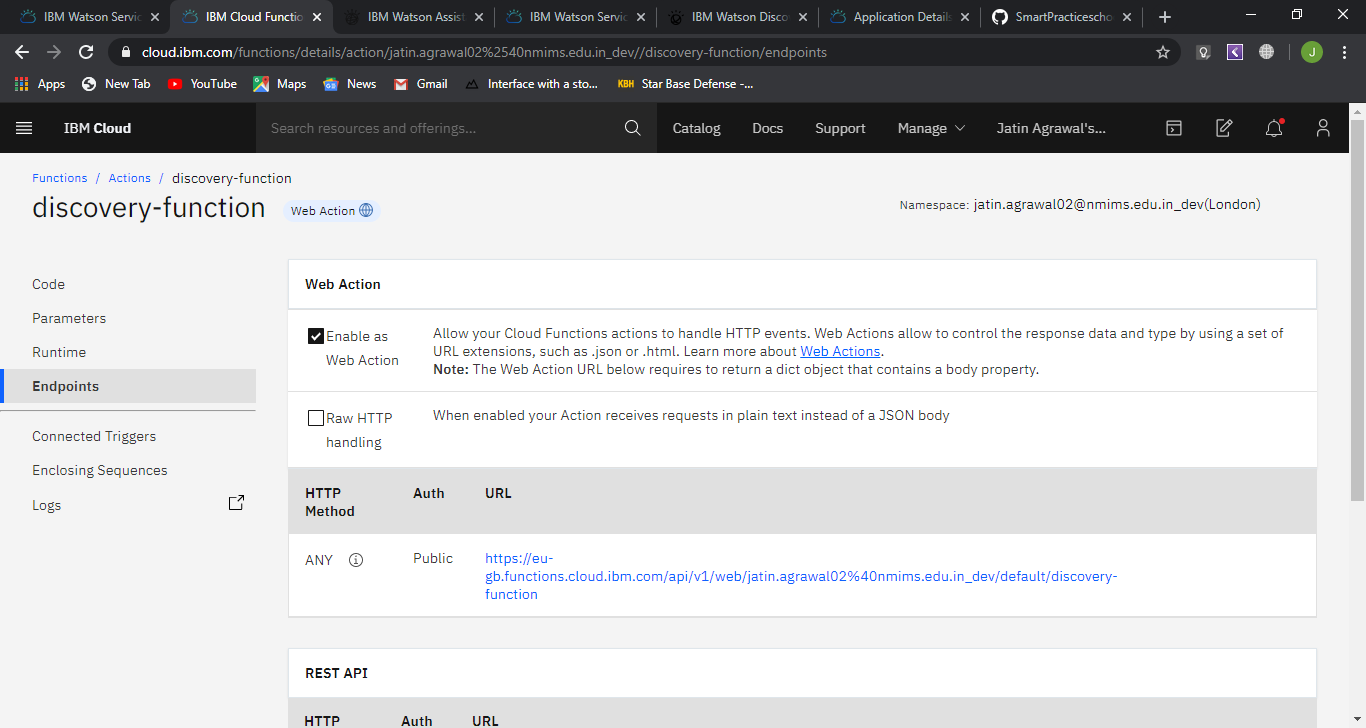


Add the following keys – url, environment\_id, collection\_id, configuration\_id and iam\_apikey.

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 Discover

Next, go to the Endpoints panel [1] 

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

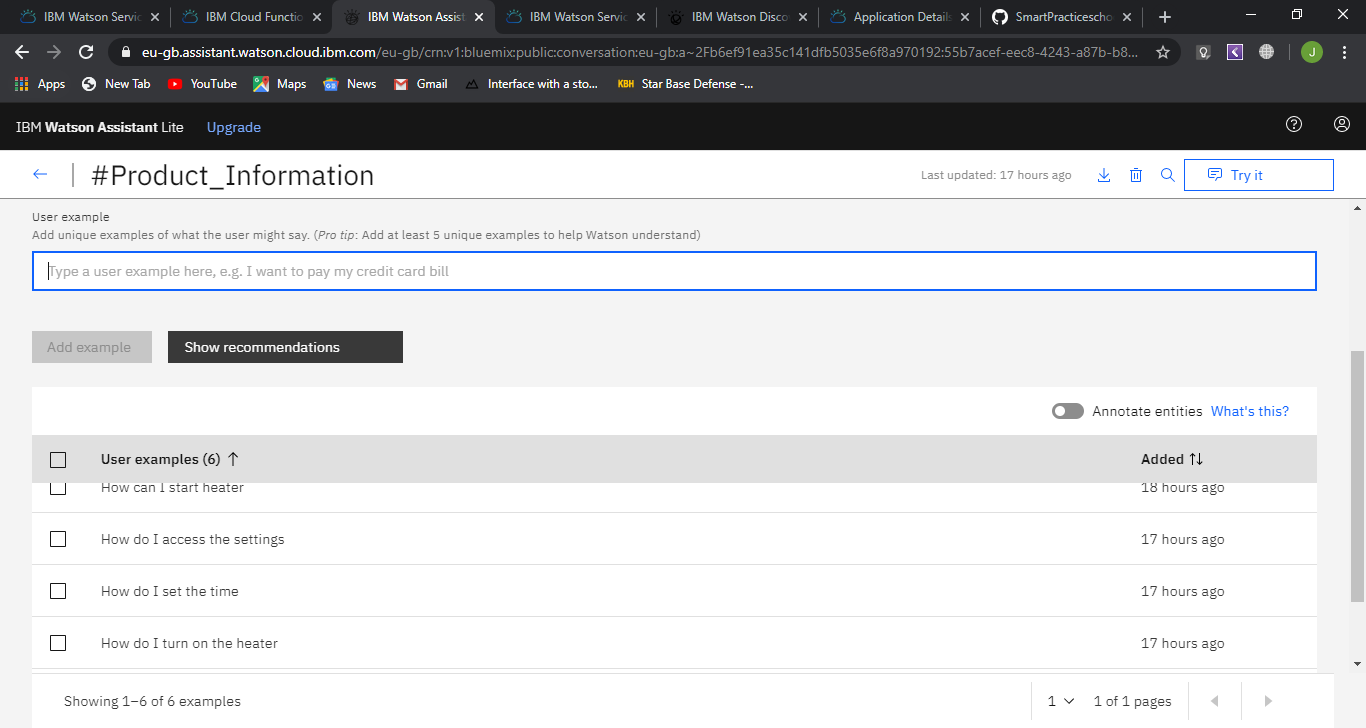
The default customer care dialog does not have a way to deal with any questions involving outside resources, so we will need to add this.

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.

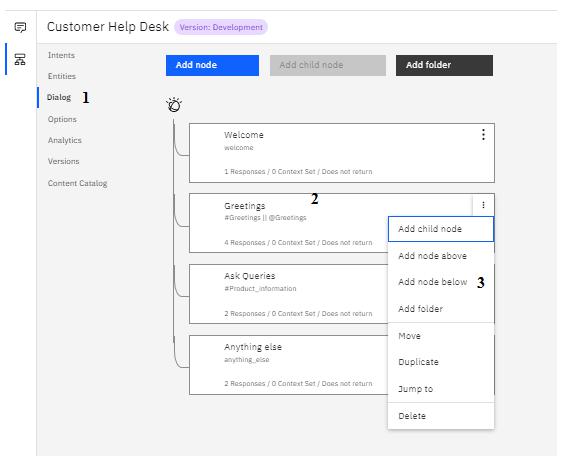
Click the Create intent button.

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

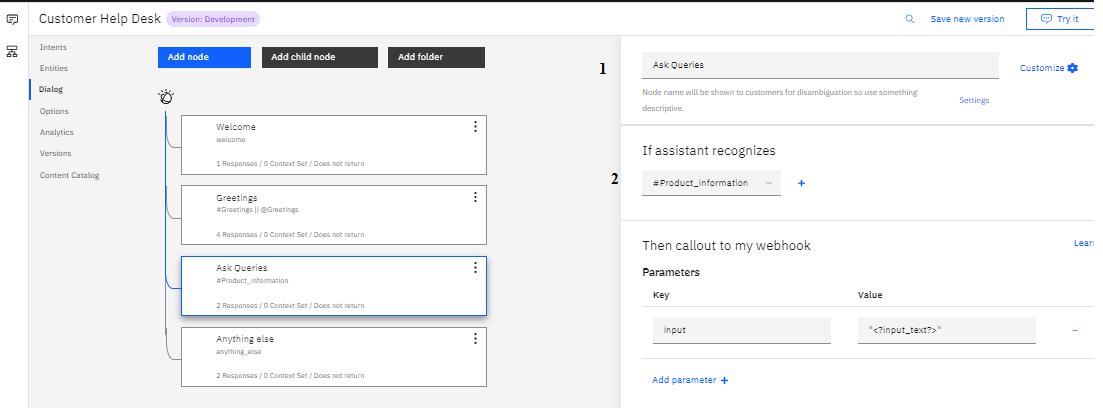


4.4.2 Create new dialog node

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].

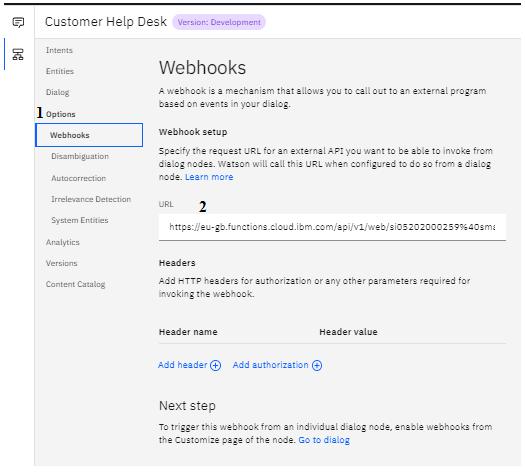


This means that if Watson Assistant recognizes a user input such as "how to flash the bios?", it will direct the conversation to this node.

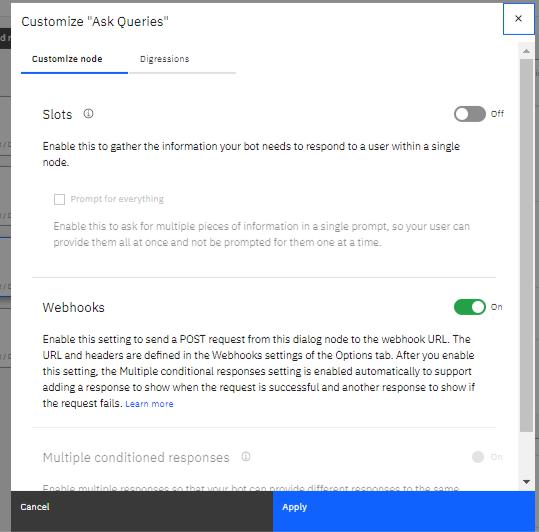
4.4.3 ENABLE WEBHOOK FROM ASSISTANT

Set up access to our WebHook for the IBM Cloud Functions action you created in Step #4.

Select the Options tab [1]:



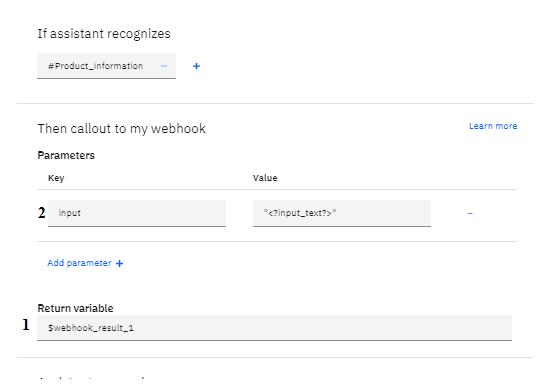
Enter the public URL endpoint for your action [2].

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:

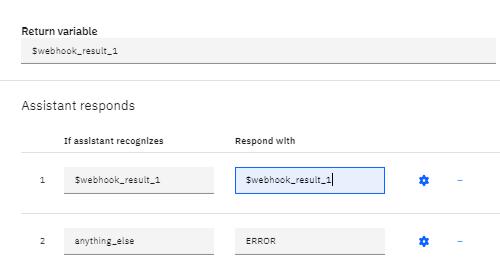
Click Apply.

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.



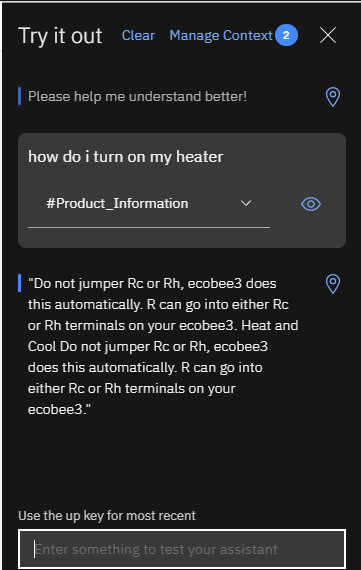
You will also need to pass in the users question via the parameter input [2]. The key needs to be set to the value: "<?input.text?>"

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:

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.

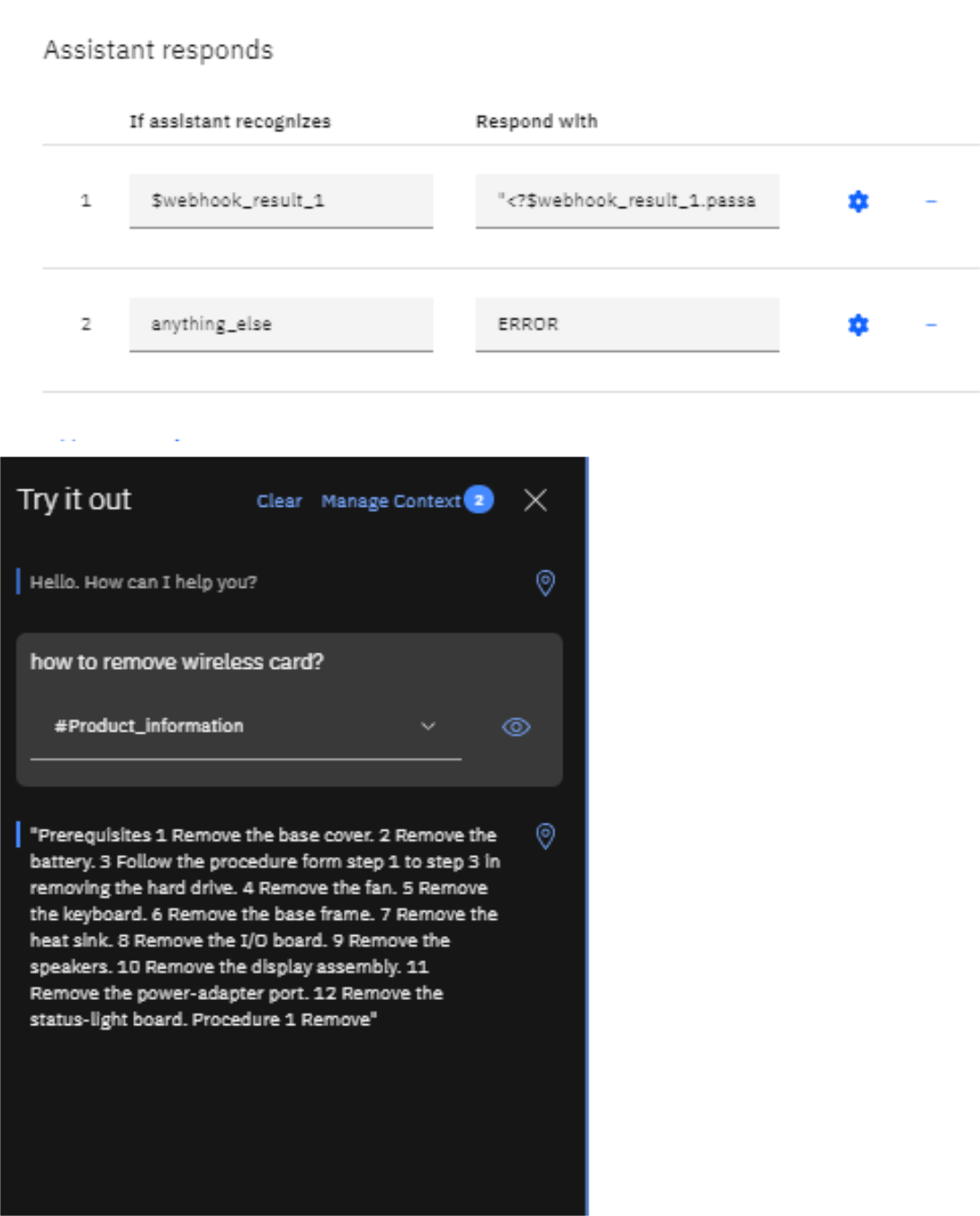
Enter some user input:



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

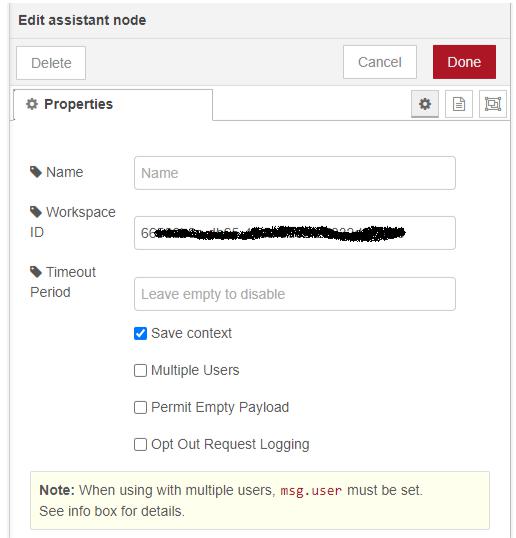
And because we specified that $webhook\_result\_1.passages be the response, that value is displayed also.

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

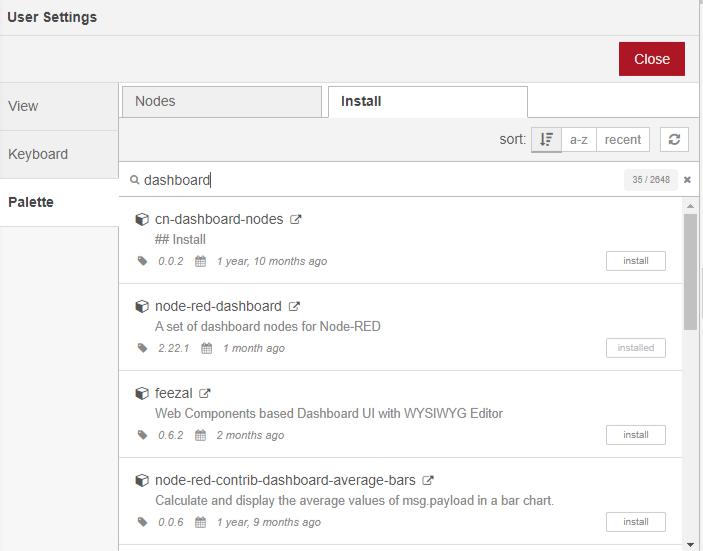


4.5 CREATE FLOW AND CONFIGURE NODE

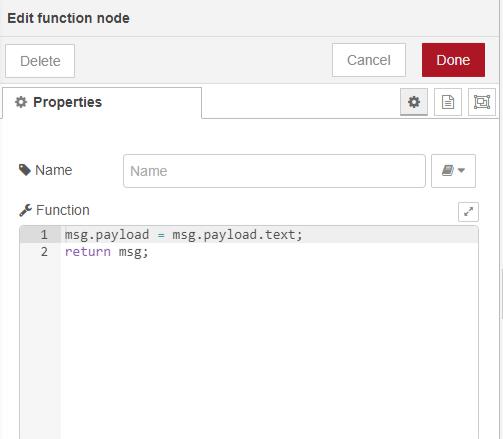
4.5.1 Integration of watson assistant in Node-RED

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

* After entering all the information click on Done.
* Drag inject node on to the flow from the Input section
* Drag Debug on to the flow from the output section
* Double-click on the inject node
* Select the payload as a string
* Enter a sample input to be sent to the assistant service and click on done
* Connect the nodes and click on Deploy
* Open Debug window as shown below
* Click on the button to send input text to the assistant node
* Observe the output from the assistant service node
* The Bot output is located inside “output.text"
* Drag the function node to parse the JSON data and get the bot response
* Double click on the function node and enter the JSON parsing code as shown below and click on done
* Connect the nodes as shown below and click on Deploy
* Re-inject the flow and observe the parsed output
* For creating a web application UI we need “dashboard“ nodes which should be installed manually.
* Go to navigation pane and click on manage palette



* 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
* Doube 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



* 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

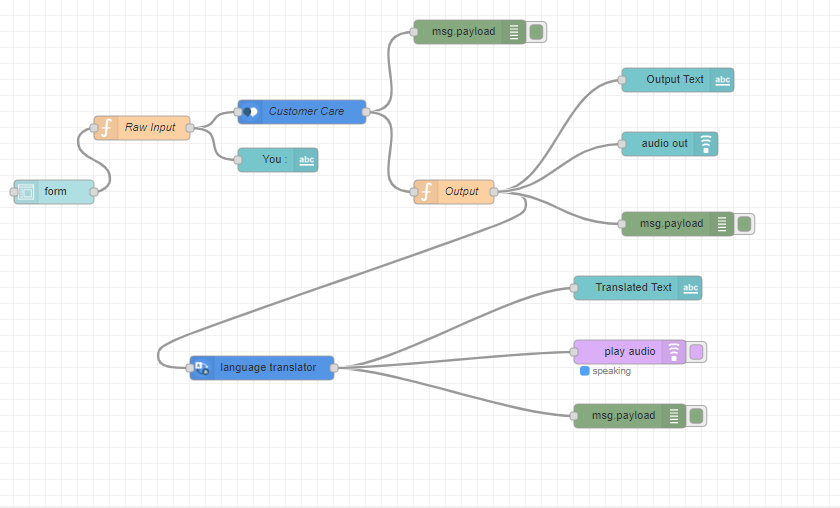
Click on Deploy.

1. **FLOWCHART**

Firstly, go to manage pallete and install dashboard.

Now create the flow with the help of following node:

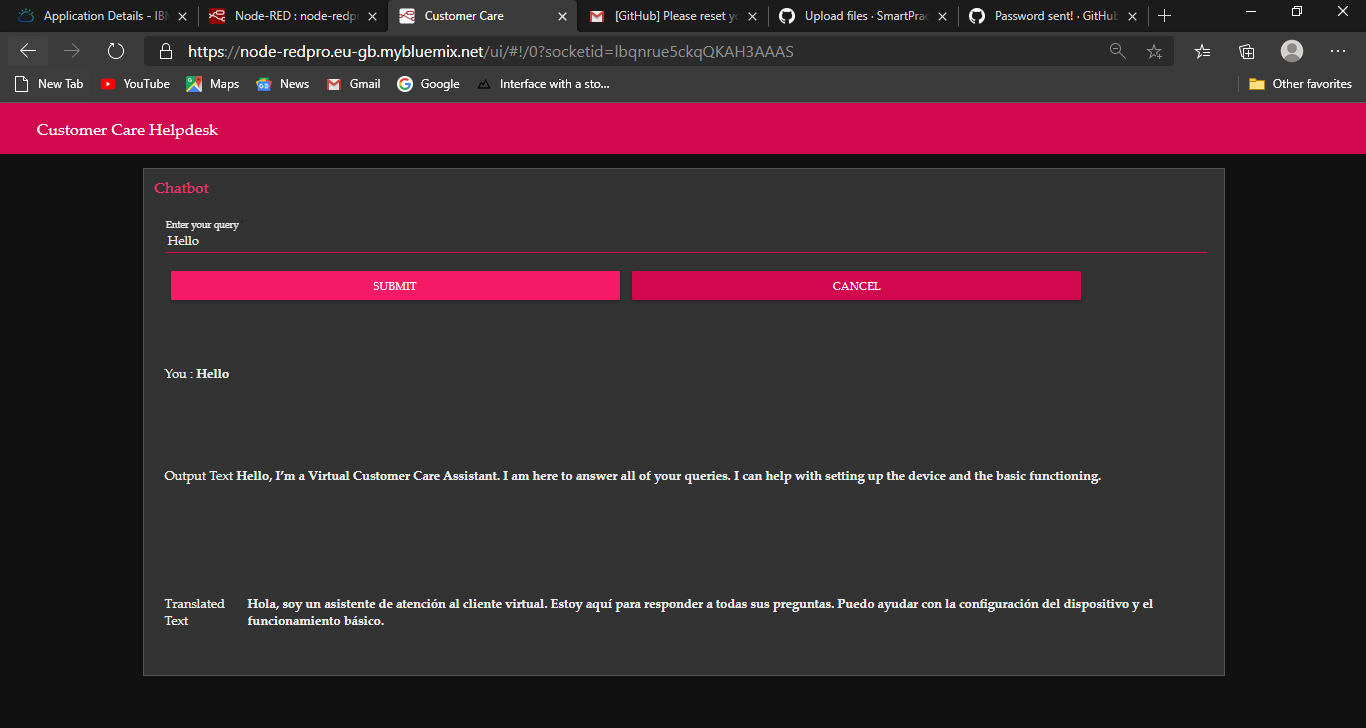
* UI\_Form
* Function
* Assistant
* Debug
* UI\_text
* Template
* Audio\_out
* Language translator
* Play audio

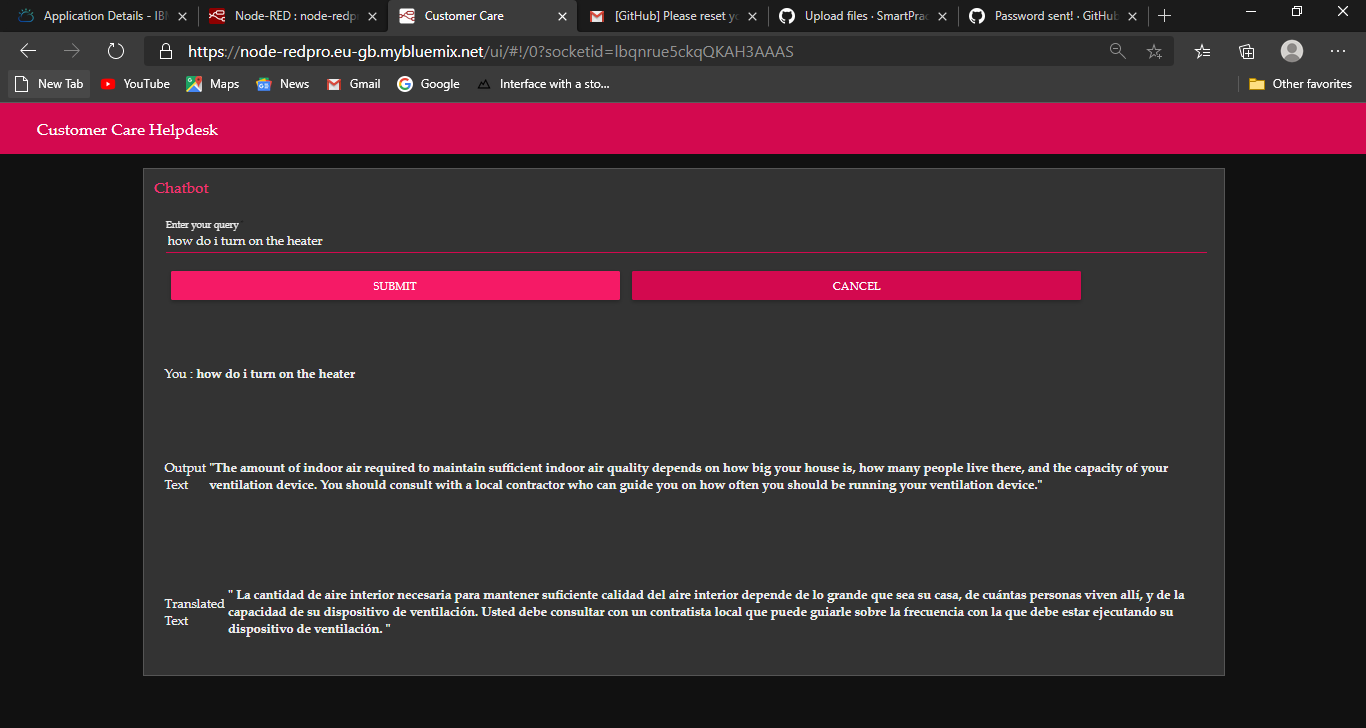


1. **RESULTS**

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.





1. **PRONS AND CONS**

PROS:

* Campanies can deploy chatbots to rectifiy 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.

1. **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.
2. **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.

1. **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.

1. **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":"c80cf8db.71f0e8","type":"tab","label":"Smart Chatbot","disabled":false,"info":""},{"id":"cb19e652.fddf58","type":"ui\_form","z":"c80cf8db.71f0e8","name":"","label":"","group":"2231fe30.fd37c2","order":1,"width":0,"height":0,"options":[{"label":"Enter your query","value":"text","type":"text","required":true,"rows":null}],"formValue":{"text":""},"payload":"","submit":"submit","cancel":"cancel","topic":"","x":70,"y":240,"wires":[["debcb84a.070b08"]]},{"id":"debcb84a.070b08","type":"function","z":"c80cf8db.71f0e8","name":"Raw Input","func":"msg.payload=msg.payload.text;\nreturn msg;","outputs":1,"noerr":0,"x":180,"y":160,"wires":[["578718b9.5f30b8","970b7d23.0c926"]]},{"id":"d560fec4.8603c","type":"function","z":"c80cf8db.71f0e8","name":" Output","func":"msg.payload=msg.payload.output.text[0];\nreturn msg;","outputs":1,"noerr":0,"x":570,"y":240,"wires":[["bcd8945c.88e838","24cf05f6.06591a","dfb70d0.32028f","452aa329.a7c77c"]]},{"id":"578718b9.5f30b8","type":"watson-conversation-v1","z":"c80cf8db.71f0e8","name":"Customer Care","workspaceid":"410577d6-bd58-4526-9abc-080bb95da53f","multiuser":false,"context":true,"empty-payload":false,"service-endpoint":"https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/55b7acef-eec8-4243-a87b-b83548511403","timeout":"","optout-learning":false,"x":380,"y":140,"wires":[["56858673.1c3e48","d560fec4.8603c"]]},{"id":"970b7d23.0c926","type":"ui\_text","z":"c80cf8db.71f0e8","group":"2231fe30.fd37c2","order":2,"width":0,"height":0,"name":"","label":"You

:","format":"{{msg.payload}}","layout":"row-left","x":350,"y":200,"wires":[]},{"id":"56858673.1c3e48","type":"debug","z":"c80cf8db.71f0e8","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","x":590,"y":40,"wires":[]},{"id":"bcd8945c.88e838","type":"ui\_text","z":"c80cf8db.71f0e8","group":"2231fe30.fd37c2","order":3,"width":16,"height":2,"name":"","label":"Output Text","format":"{{msg.payload}}","layout":"row-left","x":850,"y":100,"wires":[]},{"id":"24cf05f6.06591a","type":"watson-translator","z":"c80cf8db.71f0e8","name":"","action":"translate","basemodel":"ar-en","domain":"general","srclang":"en","destlang":"es","password":"","apikey":"QG0\_FvAQT2iY6-FWnMNWPuOV30YYWFKO0GdUME8Yeix3","custom":"","domainhidden":"general","srclanghidden":"en","destlanghidden":"es","basemodelhidden":"ar-en","customhidden":"","filetype":"forcedglossary","trainid":"","lgparams2":true,"service-endpoint":"https://api.eu-gb.language-translator.watson.cloud.ibm.com/instances/b4dfb4a9-d1ad-4cc7-be4e-67c69e248836","x":330,"y":460,"wires":[["78f5739c.8bb26c","cb7db513.ba0d58","478936c.1931dc8"]]},{"id":"78f5739c.8bb26c","type":"ui\_text","z":"c80cf8db.71f0e8","group":"2231fe30.fd37c2","order":4,"width":16,"height":2,"name":"","label":"Translated Text","format":"{{msg.payload}}","layout":"row-left","x":800,"y":360,"wires":[]},{"id":"cb7db513.ba0d58","type":"debug","z":"c80cf8db.71f0e8","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","x":790,"y":520,"wires":[]},{"id":"dfb70d0.32028f","type":"debug","z":"c80cf8db.71f0e8","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","x":850,"y":280,"wires":[]},{"id":"478936c.1931dc8","type":"play audio","z":"c80cf8db.71f0e8","name":"","voice":"15","x":790,"y":440,"wires":[]},{"id":"452aa329.a7c77c","type":"ui\_audio","z":"c80cf8db.71f0e8","name":"","group":"2231fe30.fd37c2","voice":"en-GB","always":"","x":840,"y":180,"wires":[]},{"id":"2231fe30.fd37c2","type":"ui\_group","z":"","name":"Chatbot","tab":"3bed18d6.6445b8","order":1,"disp":true,"width":16,"collapse":false},{"id":"3bed18d6.6445b8","type":"ui\_tab","z":"","name":"Customer Care Helpdesk","icon":"dashboard","disabled":false,"hidden":false}]

1. **REFERENCES**
2. <https://developer.ibm.com/articles/introduction-watson-discovery/>
3. <https://www.cxservice360.com/2018/06/27/10-interesting-applications-of-chatbots-2/>
4. <https://github.com/IBM/watson-discovery-sdu-with-assistant>
5. <https://medium.com/@rimaibrahim/node-red-watson-discovery-chatbot-telegram-ce616ddcd0d9>
6. <https://www.youtube.com/embed/Jpr3wVH3FVA>
7. <https://cloud.ibm.com/docs/openwhisk?topic=cloud-functions-getting-started>
8. <https://www.ibm.com/cloud/watson-assistant/>