

# PROJECT REPORT

**Name: Neel Raval**

**E-mail id: pnraval2007@gmail.com**

**Title: Intelligent Customer Help Desk with Smart Document Understanding – SB33721**

**Category: Artificial Intelligence Internship at *SmartInternz***

**Application ID: SPS\_APL\_20200002496**

**Project ID: SPS\_PRO\_99**

# **INDEX**

## **1. INTRODUCTION**

### **1.1 Overview**

### **1.2 Purpose**

## **2. LITERATURE SURVEY**

### **2.1 Existing Problem**

### **2.2 Proposed Solution**

## **3. THEORITICAL ANALYSYS**

### **3.1 Block Diagram**

### **3.2 Hardware and Software Designs**

## **4. EXPERIMENTAL INVESTIGATIONS**

## **5. FLOWCHART**

## **6. RESULT**

## **7. ADVANTAGES & DISADVANTAGES**

## **8. APPLICATIONS**

## **9. CONCLUSION**

## **10.FUTURE SCOPE**

# **11.BIBLOGRAPHY**

## **APPENDIX**

- a. SOURCE CODE
- b. REFERENCE

# INTRODUCTION

## 1.1 Overview

A chatbot is made to answer some really simple questions. It can answer basic questions like, “When is the store open”, “Where are you located in Koramangala” and some more simple questions.

We will enhance the chatbot to handle the queries in a better way. If the user has a very particular question, our chatbot will look for the answer in the manual provided to the Watson Discovery by us. This way we handle the queries very efficiently.

We will build a chatbot using certain functionalities of the IBM Cloud.

- ❖ Project Requirements

IBM Cloud, IBM Watson, Node-Red, NodeJS

- ❖ Functional Requirements

IBM Cloud

- ❖ Technical Requirements

Artificial Intelligence, Watson Discovery, Node-Red

- ❖ Software Requirements

Watson Assistant, Watson Discovery, Node-Red

- ❖ Project Deliverables

Intelligent Chatbot with Smart Document Understanding

- ❖ Project Team

Neel Raval

❖ Project Schedule

19 Days + 10 Days

## 1.2 Purpose

As said previously, a simple chatbot can understand and reply back to simple queries. In a case where the chatbot is unable to answer the query, it says the question is invalid or in some cases offers the user an agent to speak to.

Now, we will integrate another step into our chatbot. If the user asks a technical question, the chatbot will pass the question to our Watson Discovery, which will look for matching cases and give the reply accordingly. So, now we have moved the talking to a agent step a little further away.

We are going to use Smart Document Understanding, which is a feature of the Watson Discovery, we train the assistant accordingly, to understand, what text is important and what is not.

Hence, we improve our results.

### 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 Function's Web Action that allows Watson Assistant to post queries to Watson Discovery.
- Build a web application with integration to all these services and deploy the same on IBM Cloud Platform.

## 2. LITERATURE SURVEY

### 2.1 Existing Problem

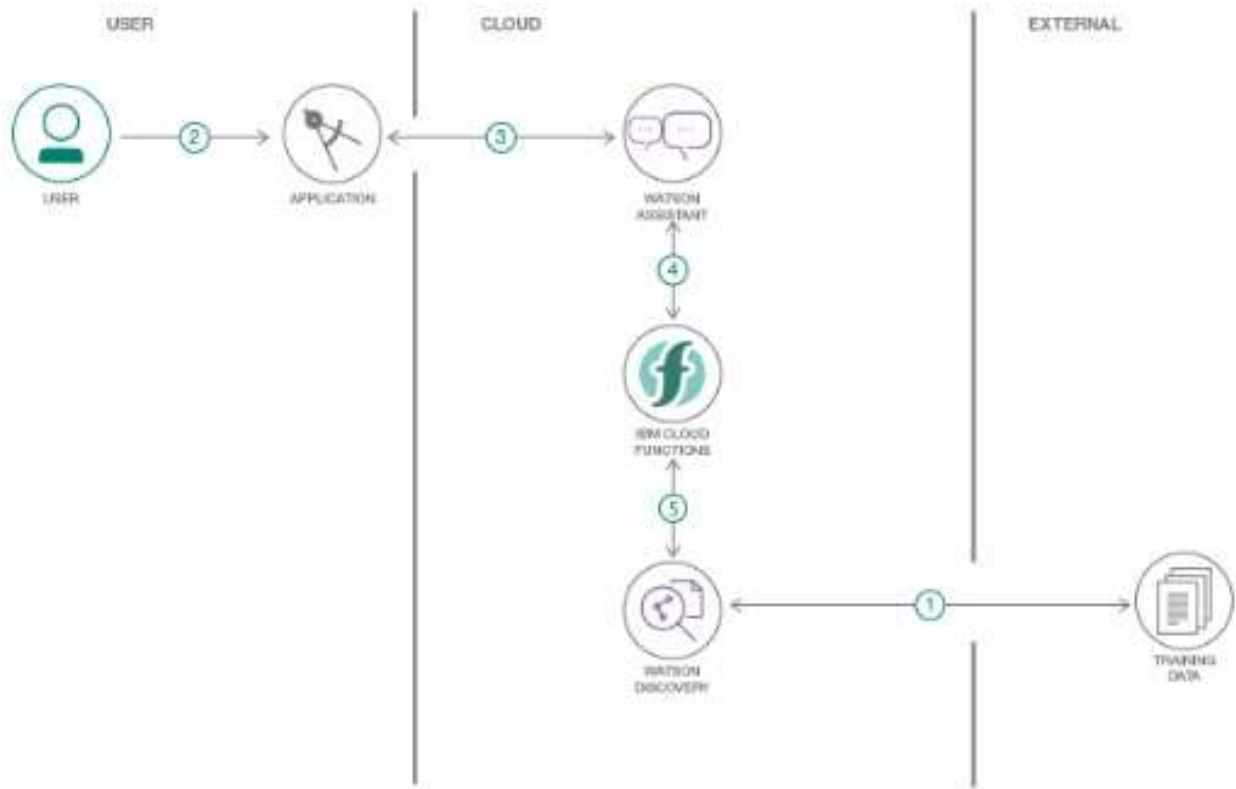
A chatbot's basic duty is to take the input query and find an appropriate and matching answer to it, when it is unable to do so, it asks the user to rephrase the query or it asks the user whether he wants to be connected to an agent. But we have built chatbots to minimize human interaction, so hence, we have to come up with a more effective solution to do so.

### 2.2 Proposed Solution

We integrate our chatbot with a Virtual Agent, for better understanding of the queries. We have to be able to train our Virtual Agent with a lot of records related to the queries we are going to provide it with. We use Watson Discovery, where we manually input the dataset and train the bot to recognize various fields. We use the Watson Assistant to build a Dialog Skill. We then use Node-Red which will serve as our web interface.

## 3. THEORETICAL ANALYSIS

### 3.1 Block / Flow Diagram



This is the basic flow of our project.

1. We train the document using SDU from Watson Discovery.
2. The user interacts with our chatbot, using the UI we created using Node-Red.
3. The created Dialog Skill coordinates the chat with the user.
4. If we get a question related to the product, we pass the query to the created cloud functions.
5. The cloud functions invoke our Dataset and hence the result is passed.

### 3.2 Hardware / Software designing:

1. IBM Cloud Services
2. Configuring our Discovery Dataset
3. Actions on the IBM Cloud Functions

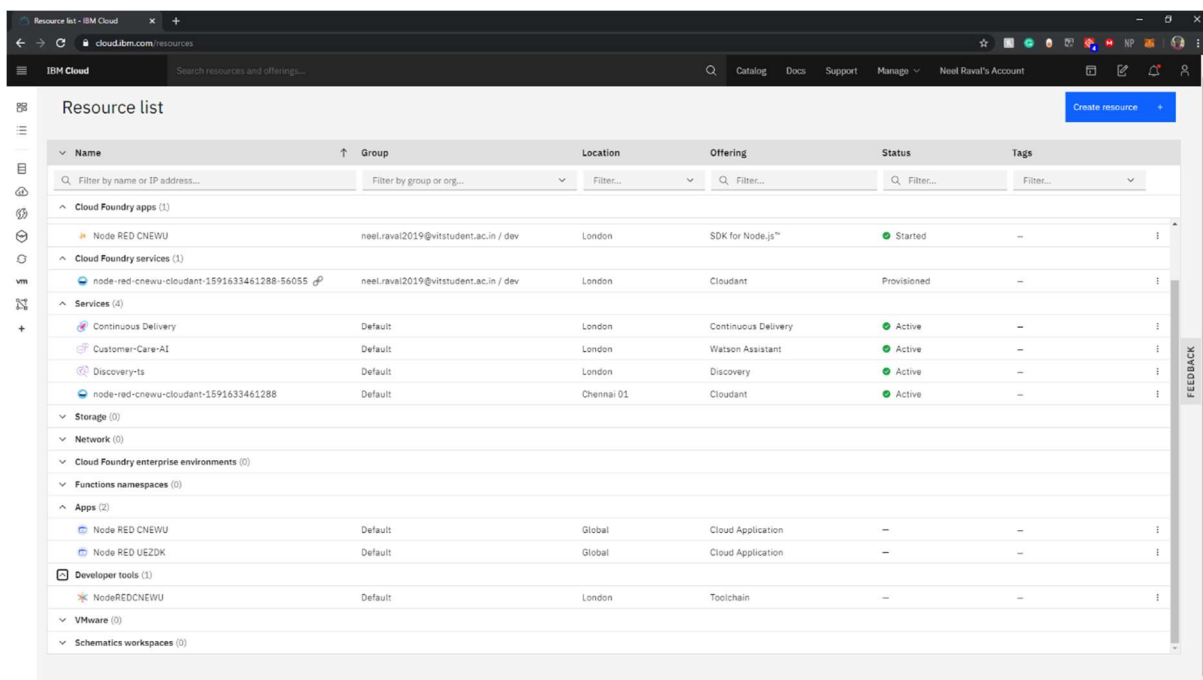
4. Watson Assistant dialog skill
5. Making the flows and configuring our dash-board flow.
6. Deployment.

## 4. EXPERIMENTAL INVESTIGATIONS

### 1. IBM Cloud Services

*Create the following Services:*

- ❖ Watson Discovery
- ❖ Watson Assistant
- ❖ Node Red



The screenshot shows the IBM Cloud 'Resource list' page. It features a table with columns: Name, Group, Location, Offering, Status, and Tags. The table is filtered to show resources for the user 'neel.raval2019@vitstudent.ac.in / dev'. The resources are categorized into groups like Cloud Foundry apps, Cloud Foundry services, Services, Storage, Network, Cloud Foundry enterprise environments, Functions namespaces, Apps, Developer tools, VMware, and Schematics workspaces. The 'Services' group is expanded, showing resources like Continuous Delivery, Customer-Care-AI, Watson Assistant, and Watson Discovery.

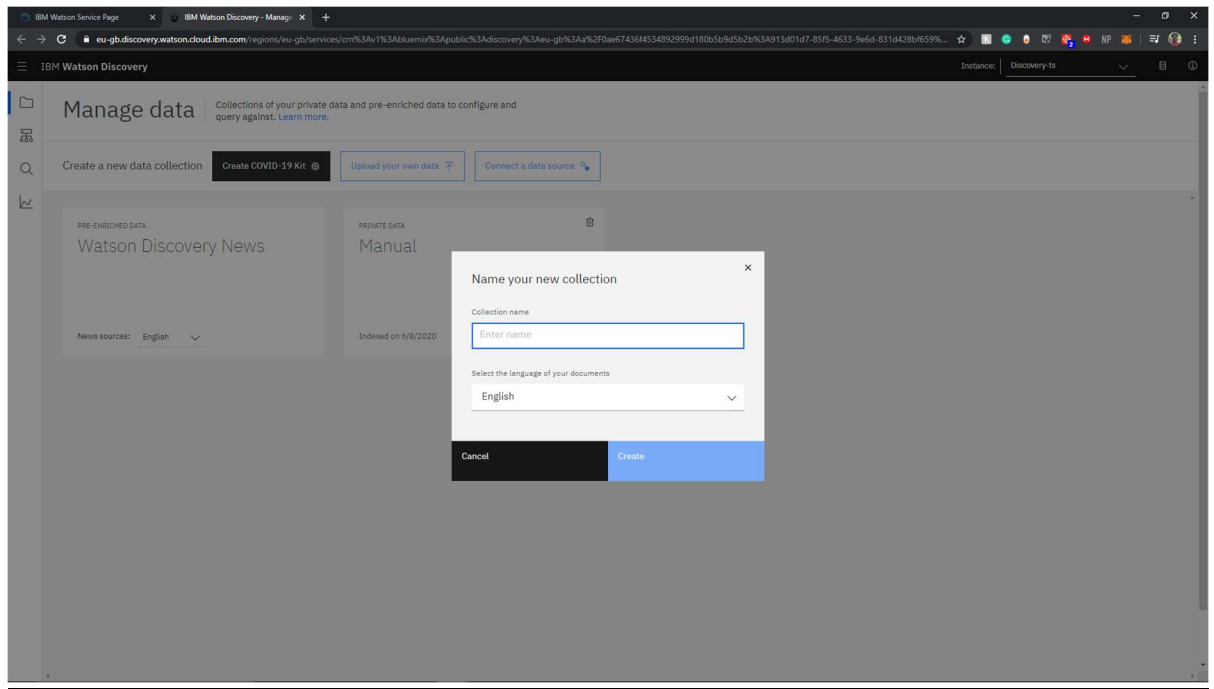
Name	Group	Location	Offering	Status	Tags
Node RED CNEWU	neel.raval2019@vitstudent.ac.in / dev	London	SDK for Node.js™	Started	—
node-red-cnewu-cloudant-1591633461288-56055	neel.raval2019@vitstudent.ac.in / dev	London	Cloudant	Provisioned	—
Continuous Delivery	Default	London	Continuous Delivery	Active	—
Customer-Care-AI	Default	London	Watson Assistant	Active	—
Discovery-ts	Default	London	Discovery	Active	—
node-red-cnewu-cloudant-1591633461288	Default	Chennai 01	Cloudant	Active	—

Upon creating the required resources you will be able to view all your resources here. Creation of all of these resources is fairly easy.

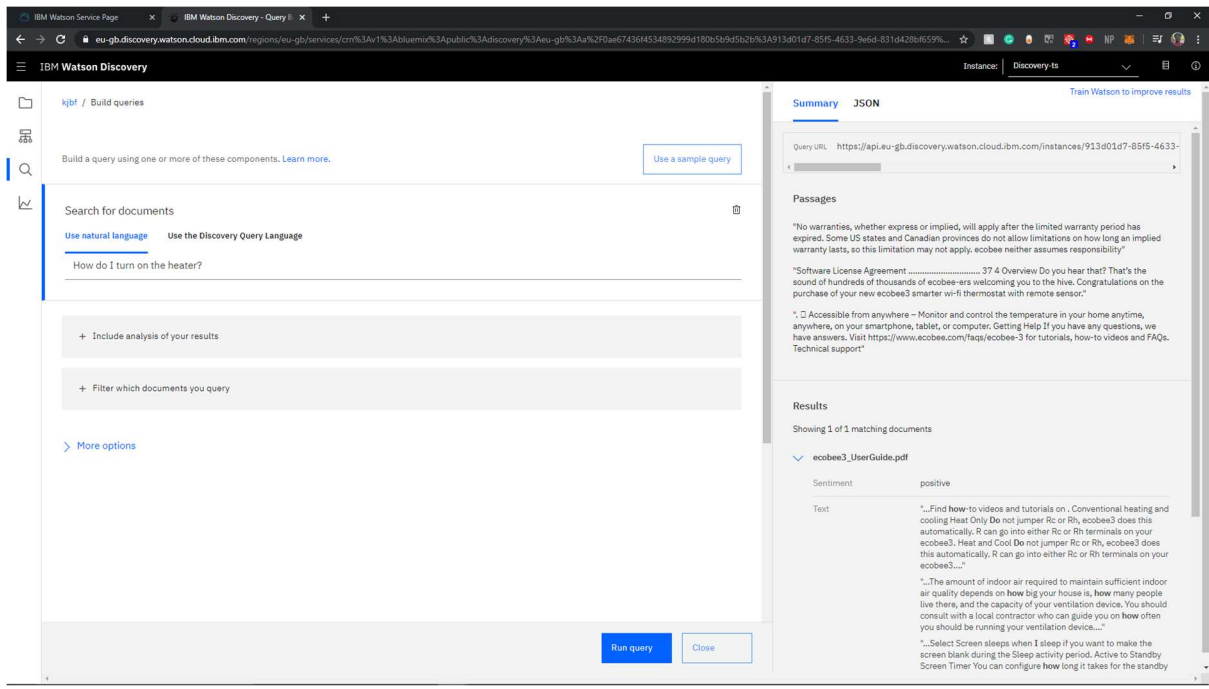
### 2. Configuring our Discovery Dataset

- ✓ Import the Document
- ✓ Create a new Data Collection

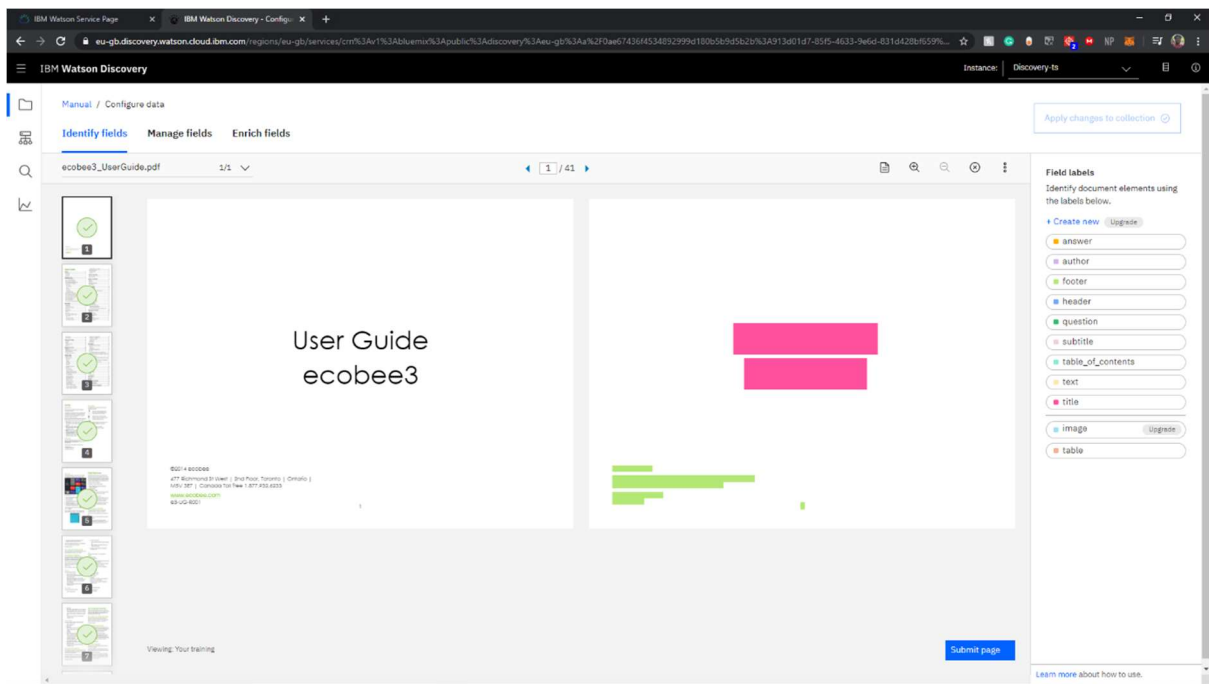




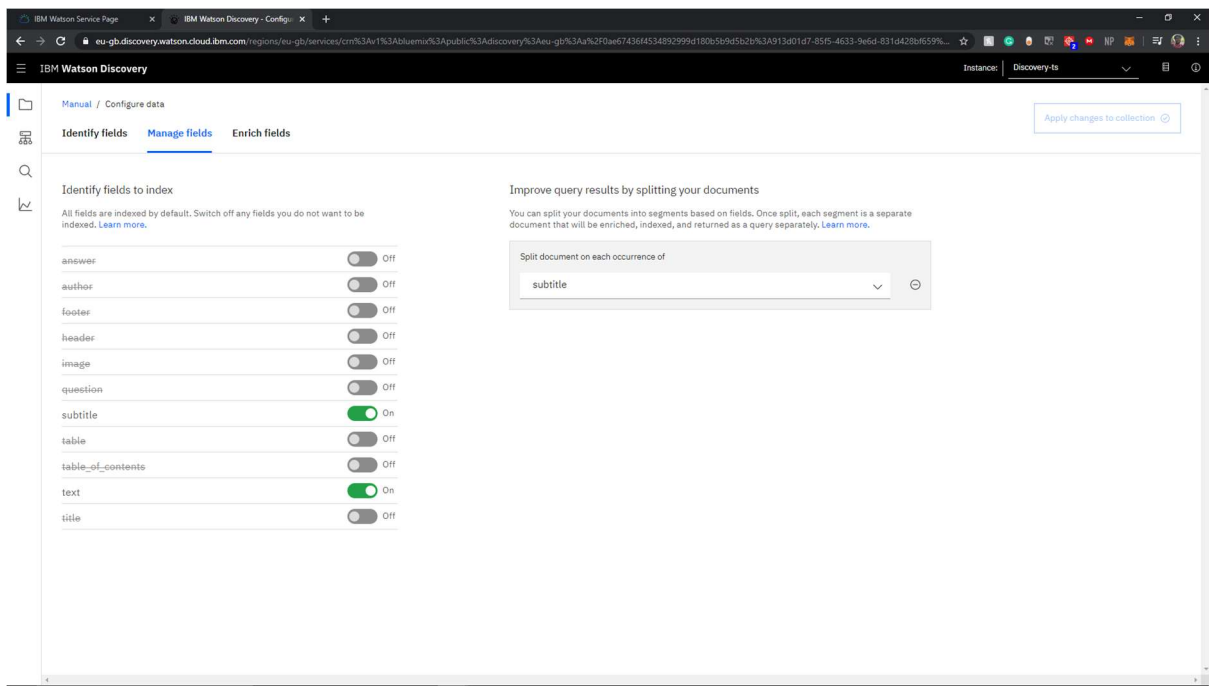
- ✓ The data collections should be given a unique name.
- ✓ Select the dataset you want to train your Watson Discovery with.
- ✓ I have uploaded Ecobee User Guide.
- ✓ Before we train our document we shall check out what our query result is.



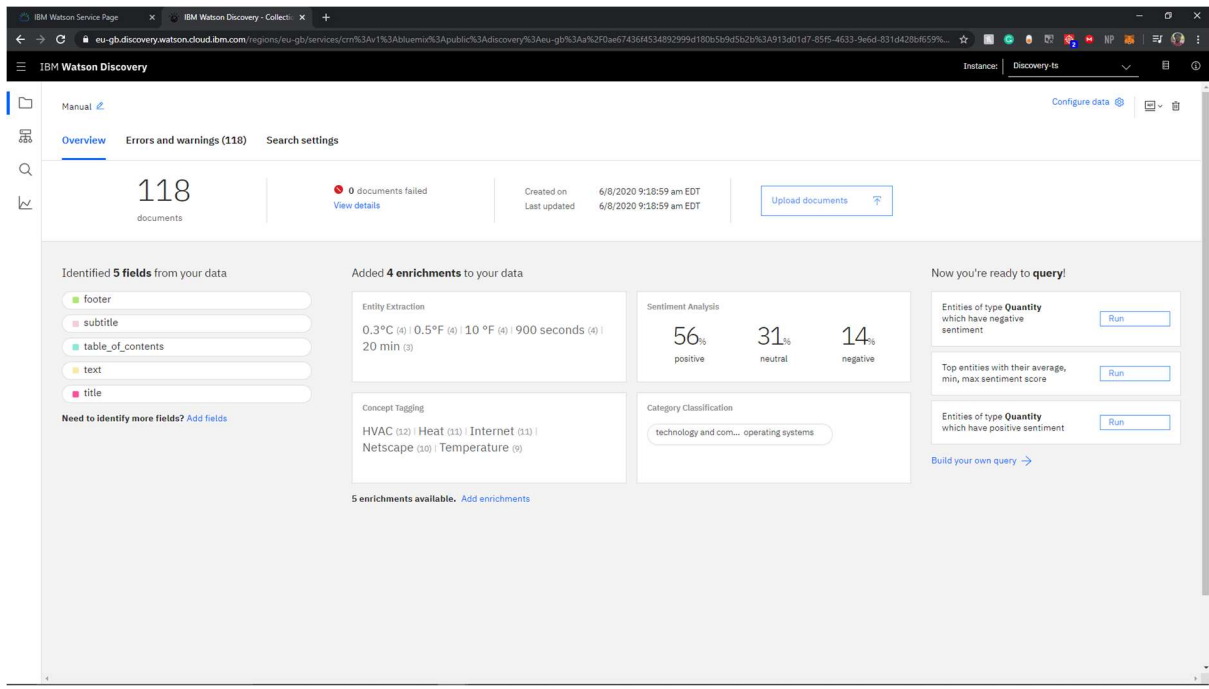
- ✓ When we run the query we see we don't get any good results.
- ✓ Now let's train the data, on the main page click on the configure data.



- ✓ Train your document according to the fields given on the right side.
- ✓ Submit all the pages and once done with all the pages, click on the Apply Changes to collection, and re-upload the dataset.
- ✓ Then let us go and Manage the fields.



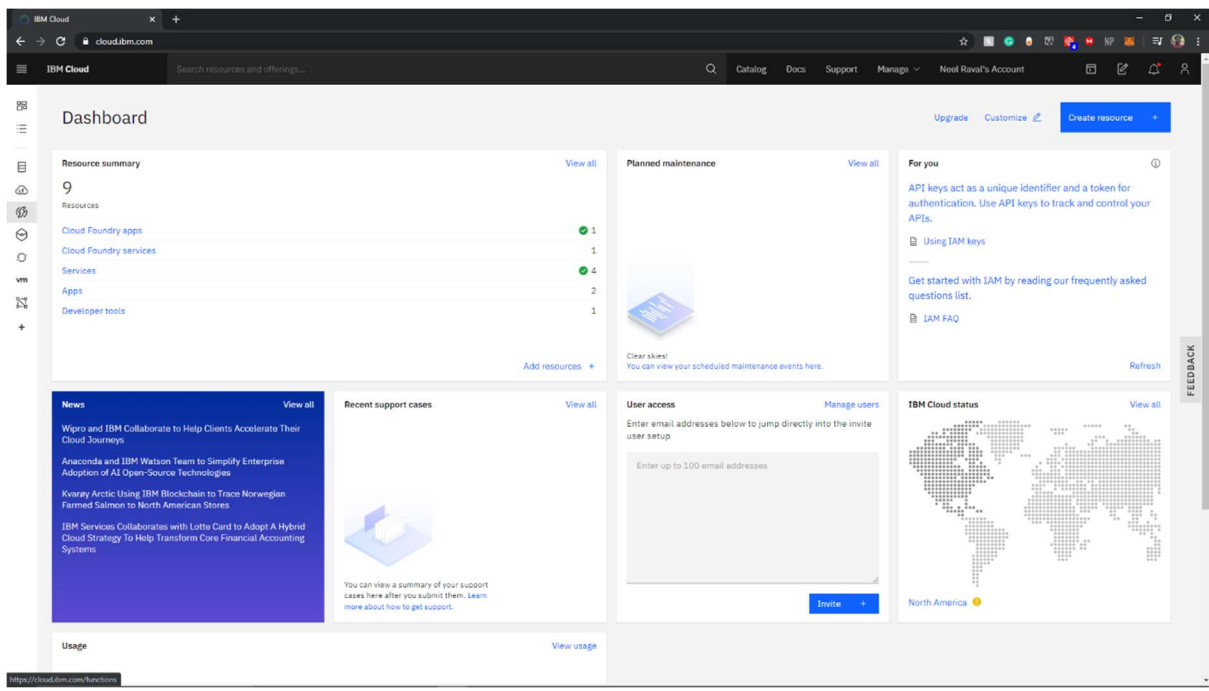
- ✓ We will tell discovery to switch off the fields that are unnecessary.
- ✓ Then we will split the document according to the subtitles we have given in the Discovery
- ✓ Once we submit our changes, we need to reupload the document.



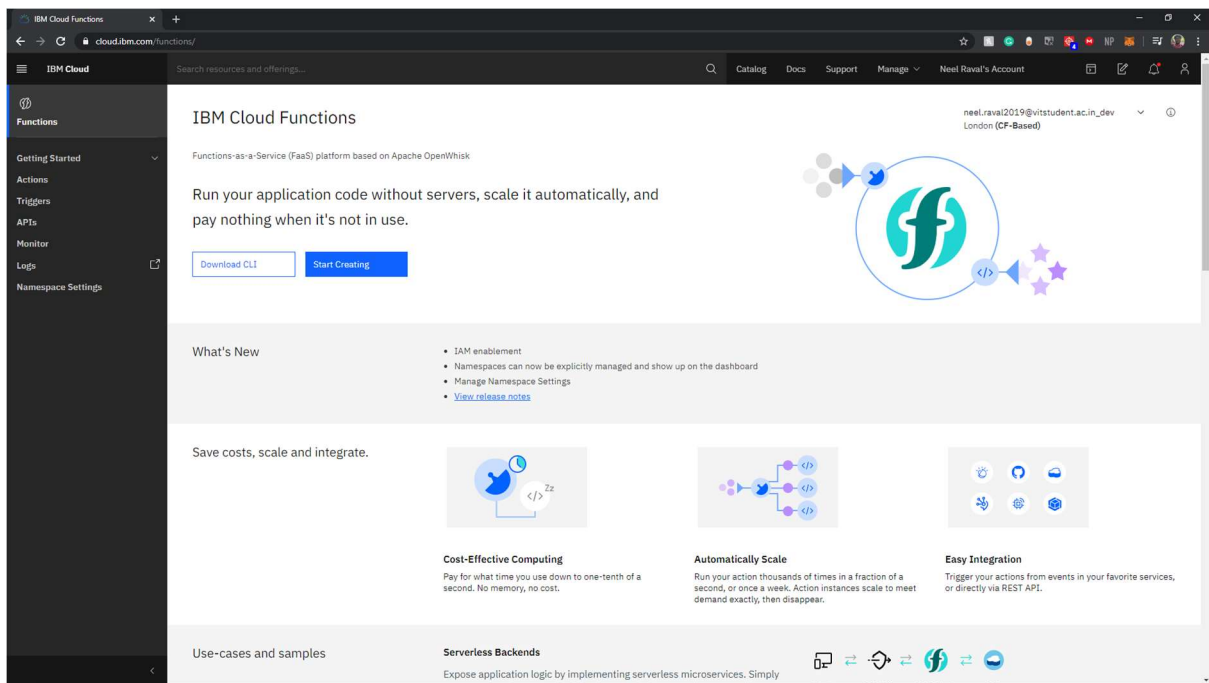
✓ As you can see, that 1 document, has been split into 118 documents. Certain enrichments have also been added to make the recognition of the document easy.

### 3. Actions on the IBM Cloud Functions

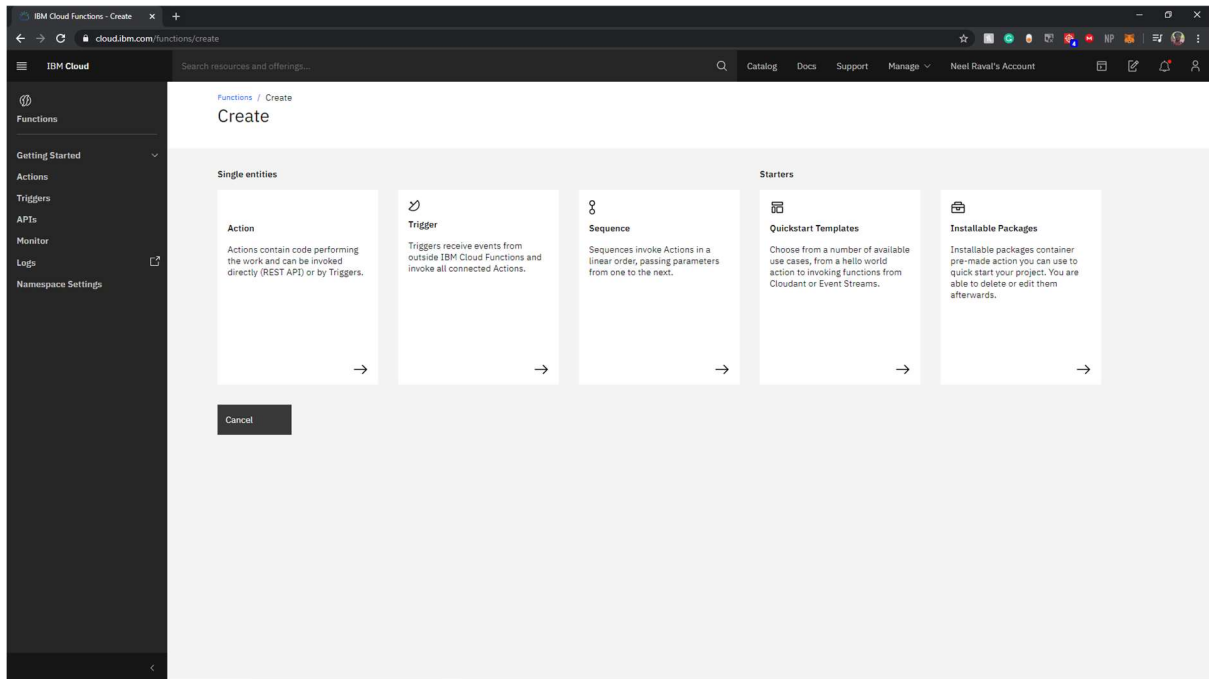
✓ We'll create the Action to make queries against Discovery.



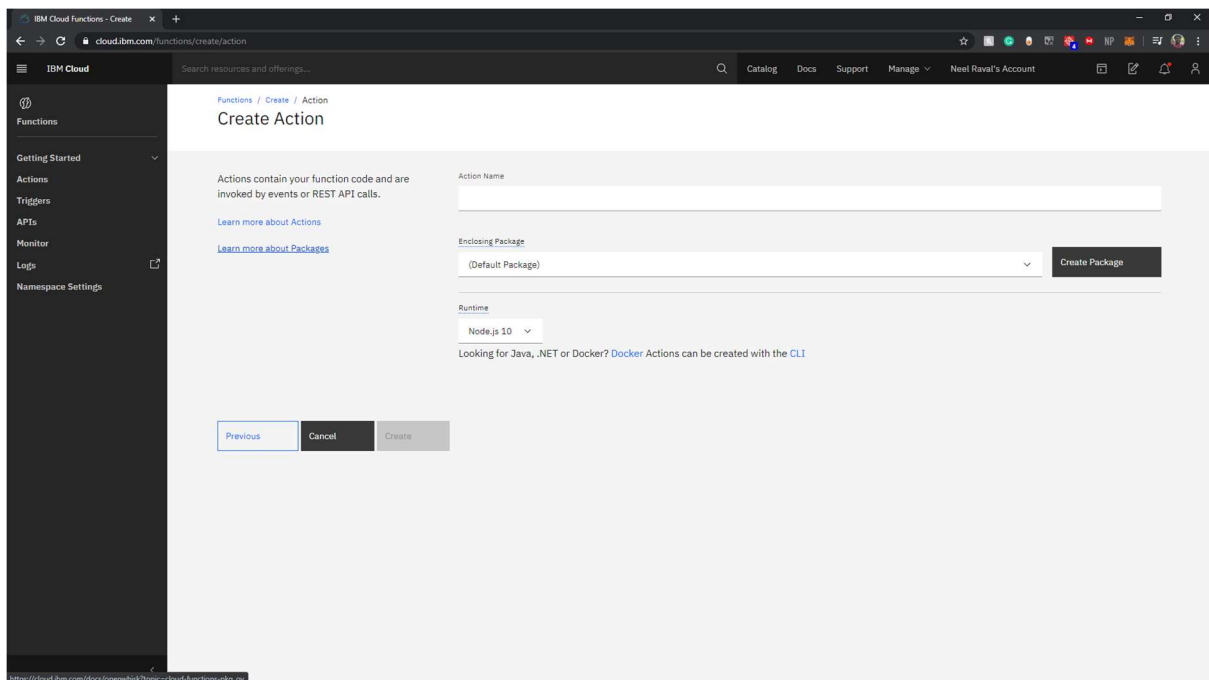
✓ As you can see, I have highlighted the function button, click on it and you'll land on the next page.



✓ This is main page of the cloud functions. IBM allows 10,000 API calls a month.



✓ Create an Action.



✓ Fill the fields accordingly and create your Cloud Actions.

The screenshot shows the IBM Cloud Functions console for the 'SDU-Smart' Web Action. The 'Code' tab is selected, displaying a Node.js 10 function. The code is a JavaScript function that takes parameters and returns a JSON object. It uses the 'assert' module and the 'DiscoveryV1' class from the 'watson-developer-cloud/discovery/v1' package. The function is named 'main' and is wrapped in a Promise. The code is as follows:

```
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.
27 // *
28 // */
29
30 function main(params) {
31   return new Promise(function (resolve, reject) {
32     let discovery;
33
34     if (params.iam_apikey) {
35       discovery = new DiscoveryV1({
36         'iam_apikey': params.iam_apikey,
37         'url': params.url,
38         'version': '2019-03-25'
39       });
40     }
41     else {
42       discovery = new DiscoveryV1({
43         'username': params.username,
44         'password': params.password,
45         'url': params.url,
46         'version': '2019-03-25'
47       });
48     }
49   });
50 }
```

✓ This is the code for the Cloud Action.

The screenshot shows the IBM Cloud Functions console for the 'SDU-Smart' Web Action, with the 'Parameters' tab selected. The parameters are listed in a table with columns for 'Parameter Name' and 'Parameter Value'. The parameters are:

Parameter Name	Parameter Value
url	
environment_id	
collection_id	
configuration_id	
iam_apikey	

✓ Fill in these details from the Discovery API details.

✓ You need to fill in the details so that when we invoke it, it gives us an appropriate response.

The screenshot shows the IBM Cloud Functions console for the 'SDU-Smart' action. The left sidebar contains a menu with 'Code', 'Parameters', 'Runtime', 'Endpoints', 'Connected Triggers', 'Enclosing Sequences', and 'Logs'. The 'Code' tab is selected, displaying a JavaScript function named 'main' that uses the 'discovery' API to search for 'heating' and 'furnace' resources. The 'Invoke with parameters' button is visible. On the right, the 'Activations' tab shows a single activation with an ID of '2437364c4084780b764c408278d61', a duration of 848 ms, and a timestamp of 10/6/2020, 15:19:44. The 'Results' section displays a JSON object containing 'results' and 'results' arrays.

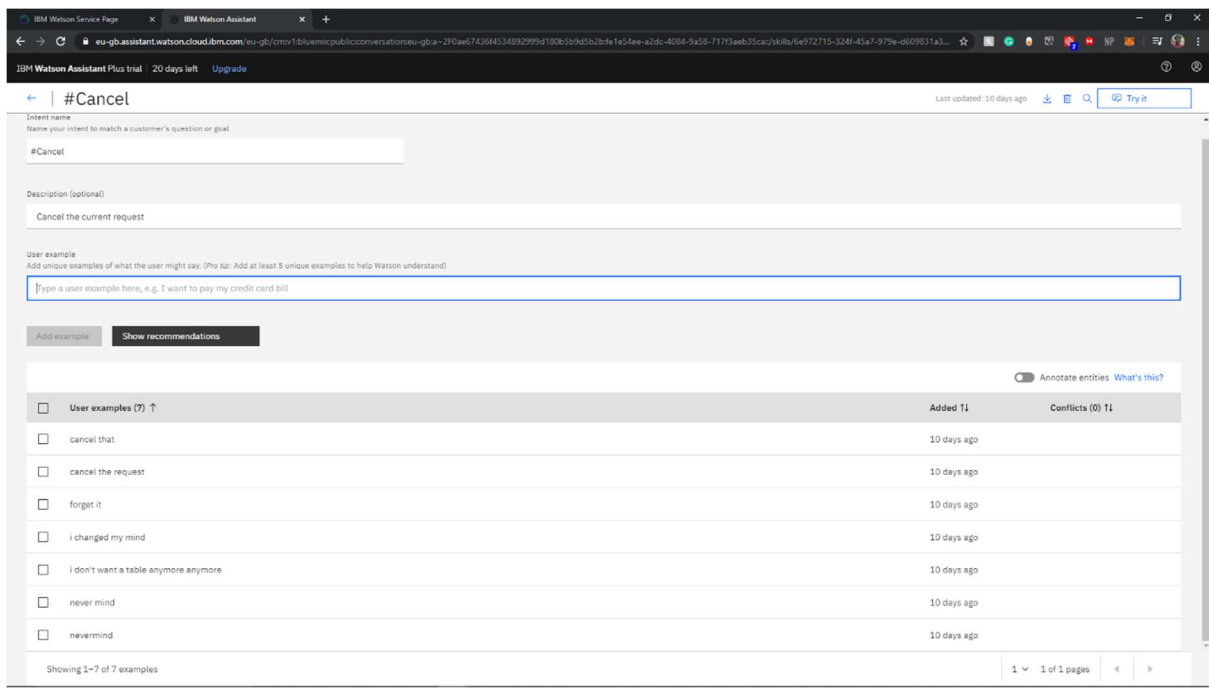
The screenshot shows the IBM Cloud Functions console for the 'SDU-Smart' action, specifically the 'Endpoints' tab. The 'Web Action' section is expanded, showing that the action is enabled as a Web Action. The 'HTTP Method' is 'ANY', the 'Auth' is 'Public', and the 'URL' is 'https://eu-gb.functions.cloud.ibm.com/api/v1/web/nee.raval2019%40vitstudent.ac.in\_dev/default/SDU-Smart'. The 'REST API' section shows a 'POST' method with 'API-KEY' authentication and a URL of 'https://eu-gb.functions.cloud.ibm.com/api/v1/namespaces/nee.raval2019%40vitstudent.ac.in\_dev/actions/SDU-Smart'. The 'CURL' section provides a curl command to invoke the action.

✓ Enable as Web Action, you'll get a public endpoint URL.

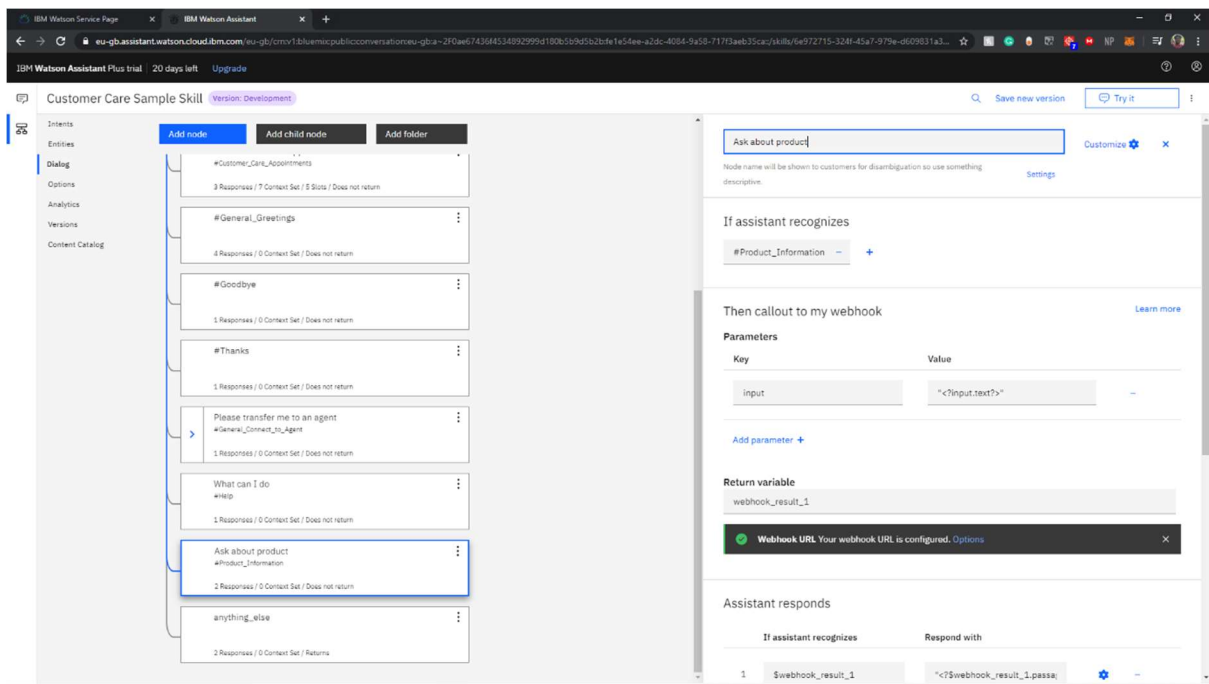
## 4. Watson Assistant Dialog Skill

✓ Create a Dialog Skill using Watson.

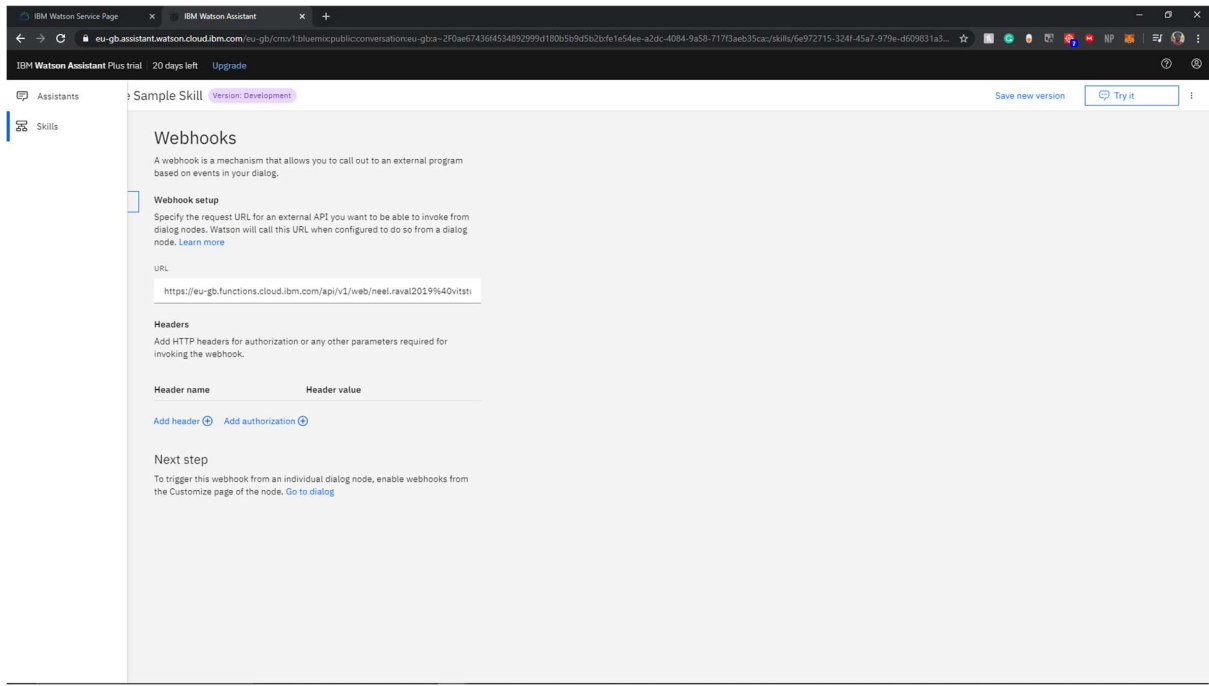




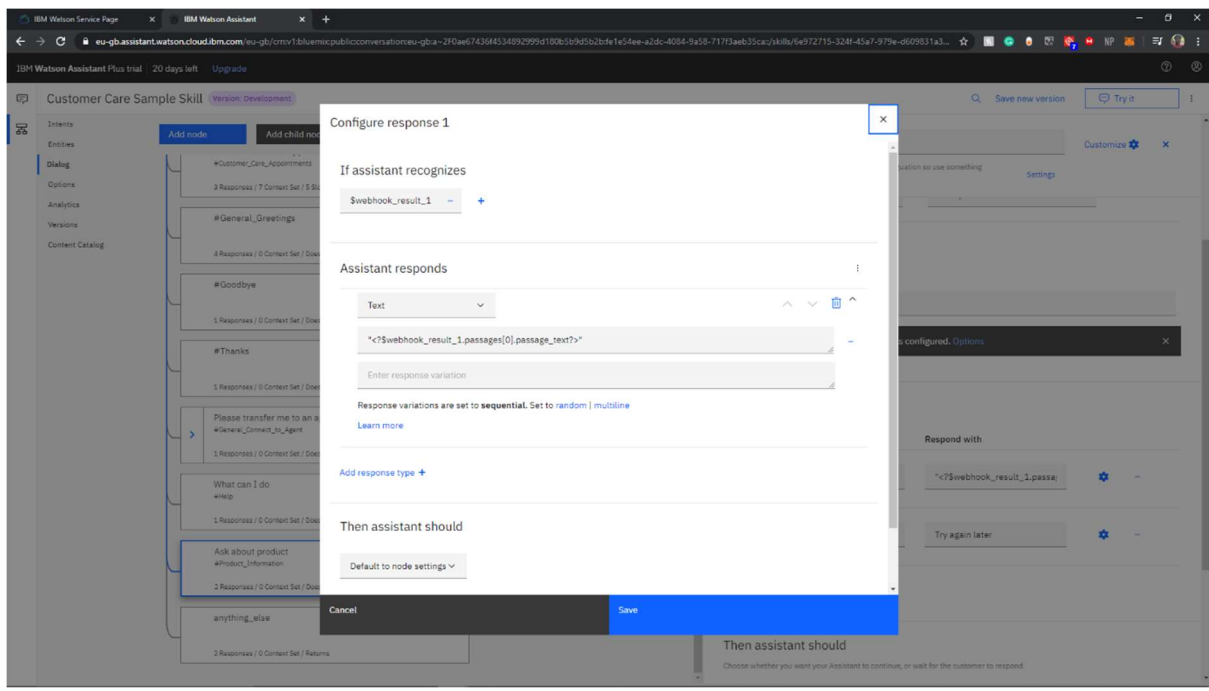
- ✓ Create an intent, fill in the intent name, the description and few examples related to it.
- ✓ We have to add as many intents as possible so that all of the queries fall into the intent.



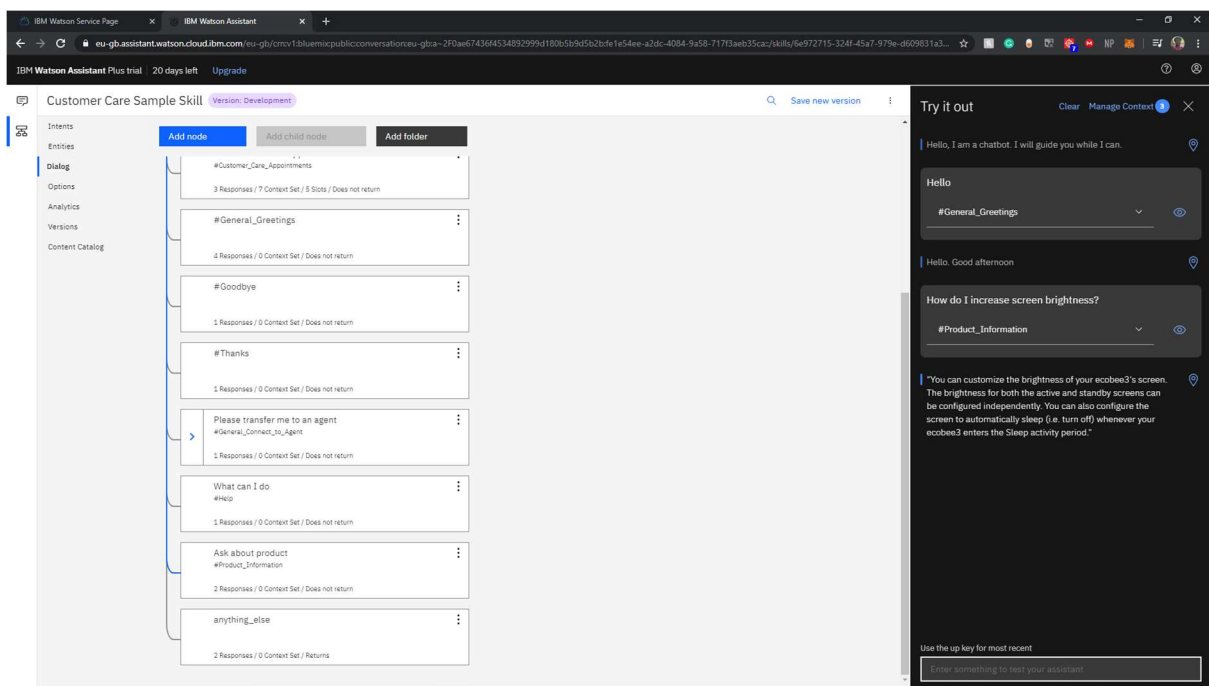
- ✓ Add a dialog node, name the node appropriately, give an intent, for recognition.



- ✓ Enable webhooks, and put the URL from the Action.
- ✓ For all the nodes as per required, enable webhooks.

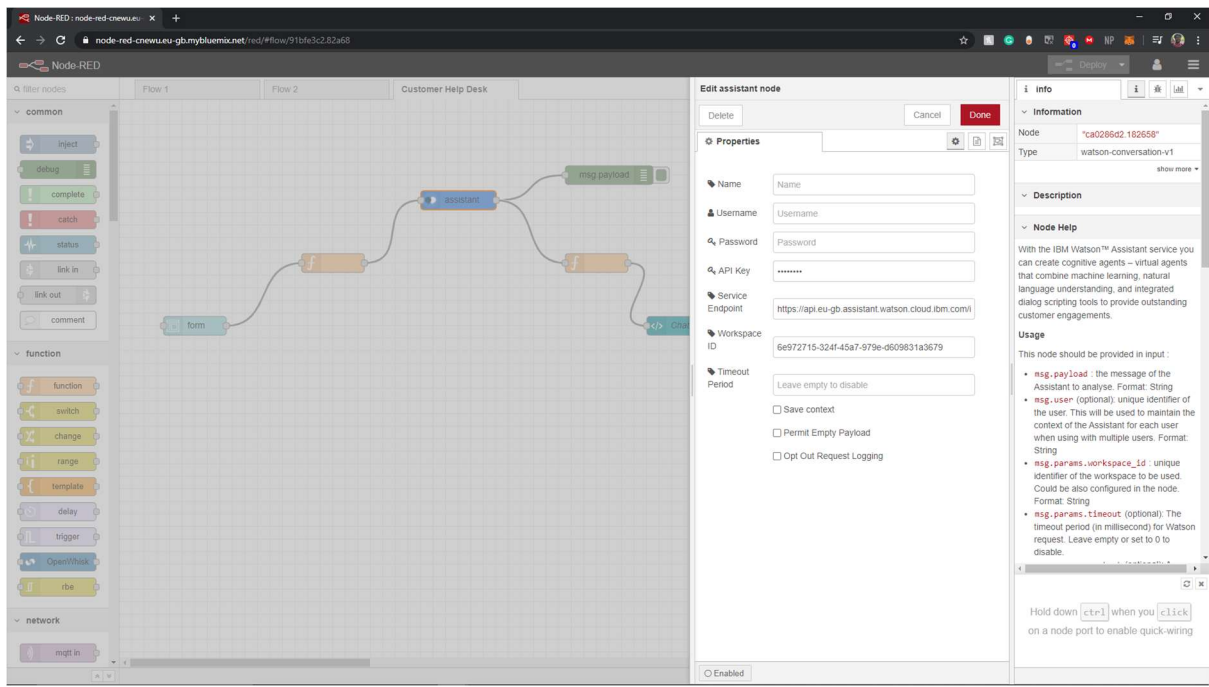


✓ Edit the webhooks and change the input-output accordingly.

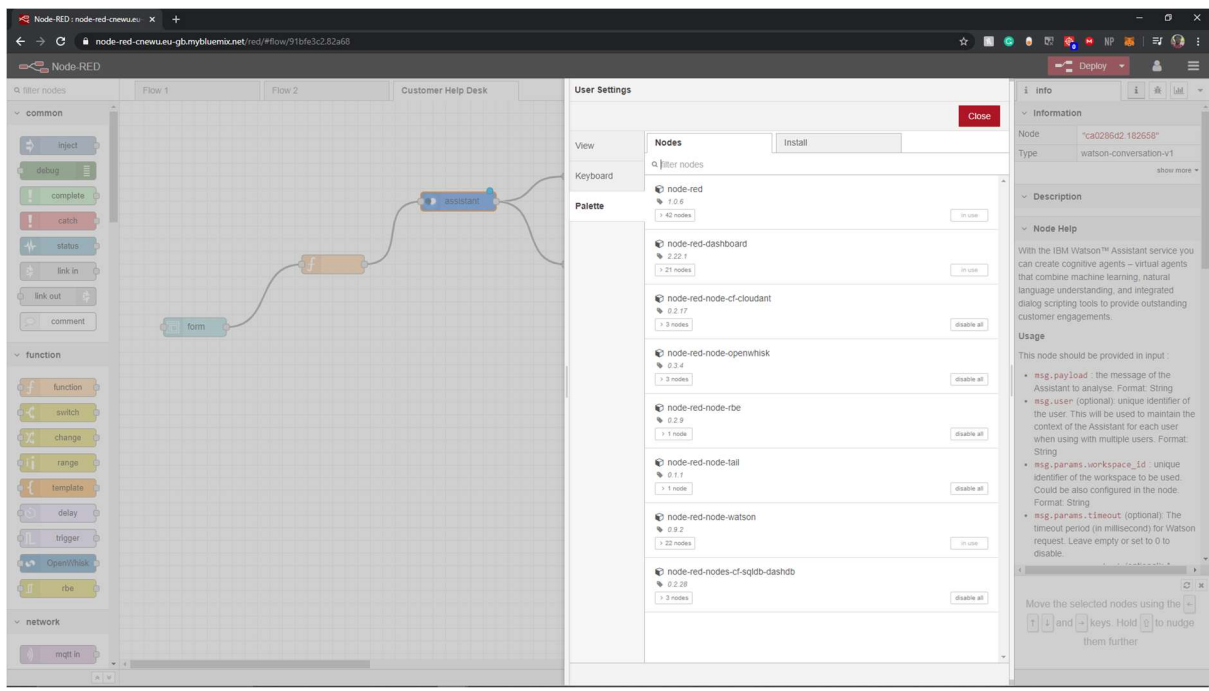


- ✓ This is the try it out window, as you can see, it replies appropriately and also tells us from which intent, it has taken the input.

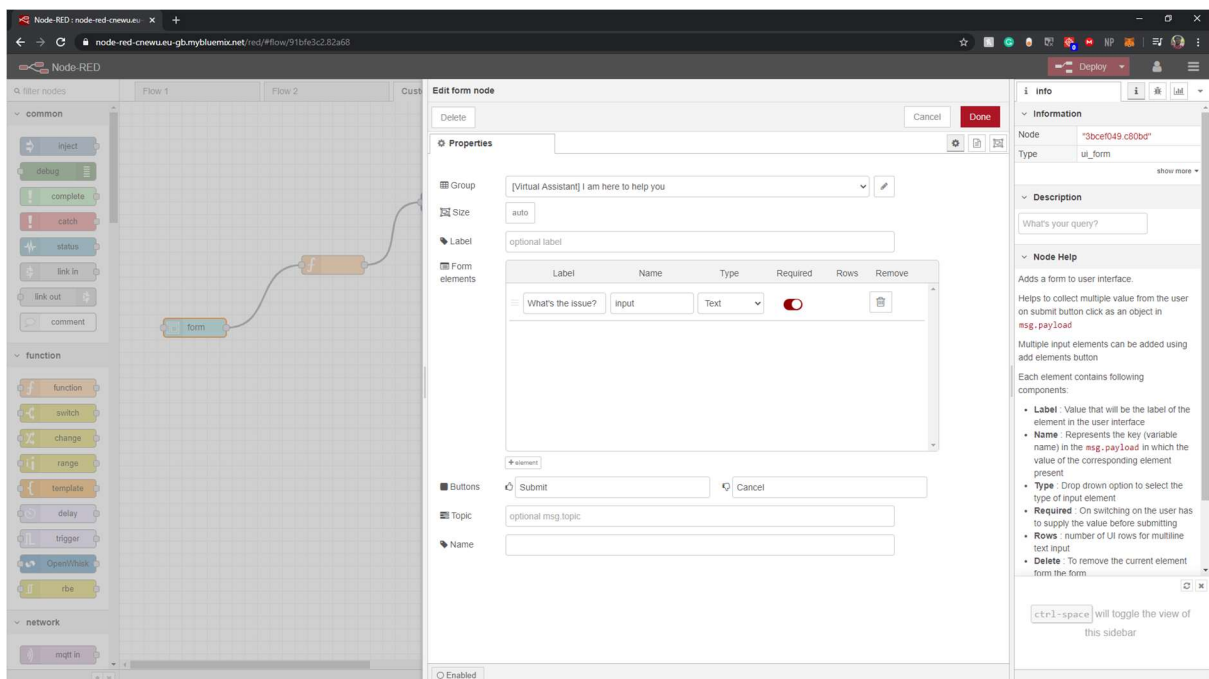
## 6. Making the flows and configuring our dash-board flow.



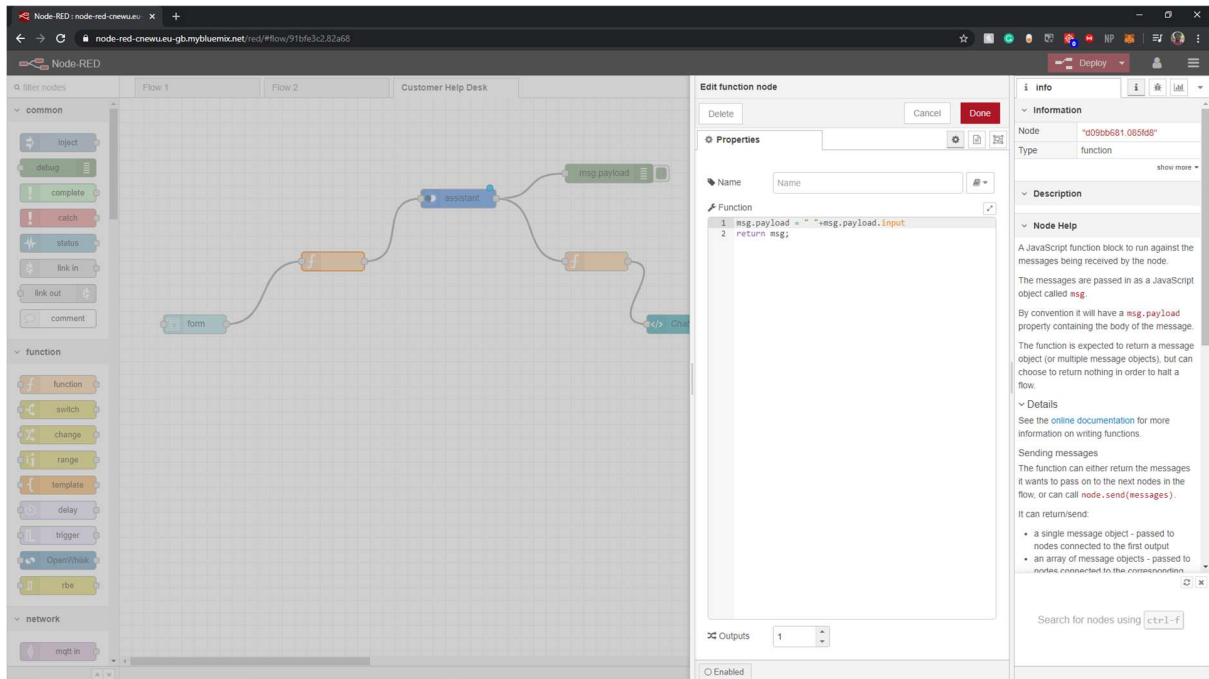
- ✓ Integrate Watson Assistant with Node-Red
- ✓ In the assistant node, fill in the required details.
- ✓ From the input section, drag the template node.
- ✓ From the output section, drag the output flow.
- ✓ Now, we'll need to install the dashboard, for the web application UI.



- ✓ Click on the hamburger menu, click on Manage Pallet, in install, search for node-red dashboard, and install it.
- ✓ Drag the form node onto the dashboard
- ✓ Configure the form

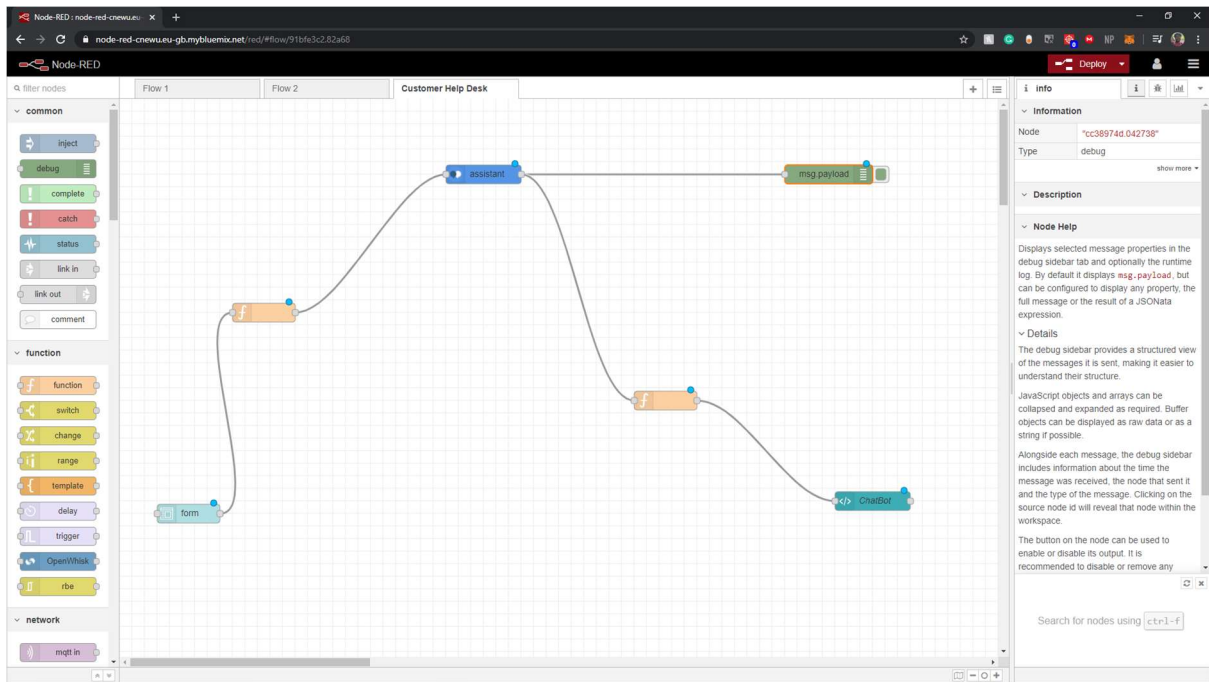


- ✓ Fill the details accordingly.
- ✓ Put in a function node.



- ✓ Connect the form output to the input of the function, and the output to the input of the assistant bot.
- ✓ Drag the text and template node onto the dashboard.
- ✓ Then Deploy the dashboard.

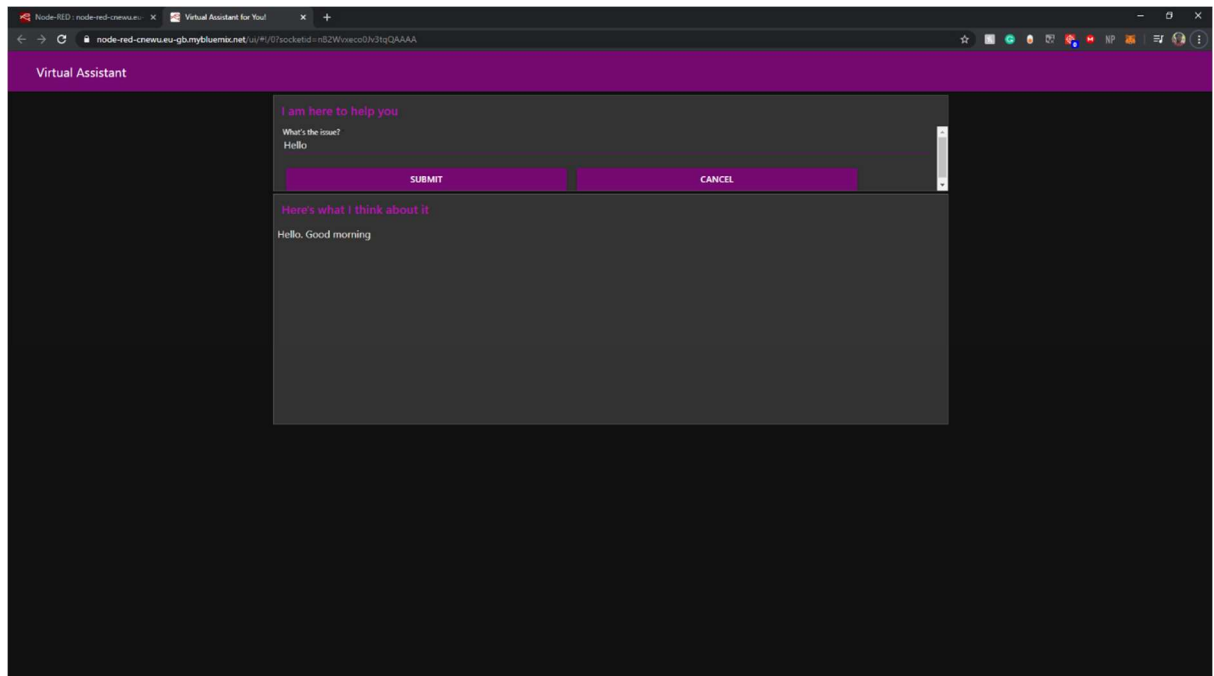
## 5. FLOWCHART

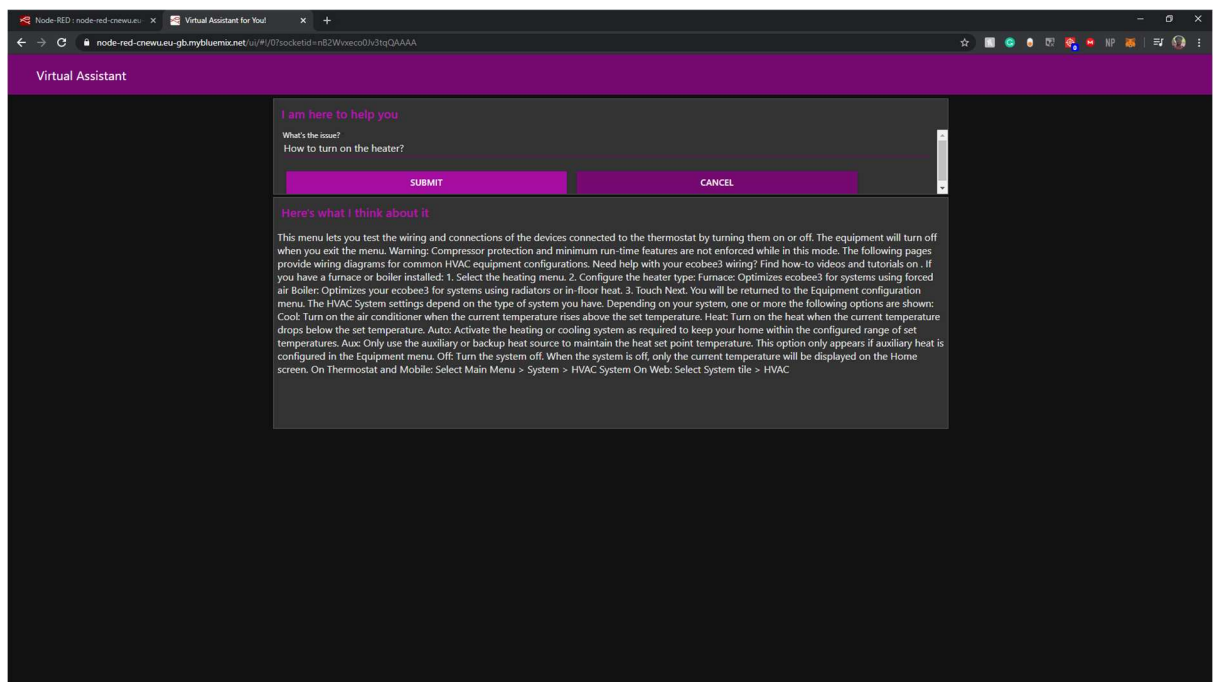
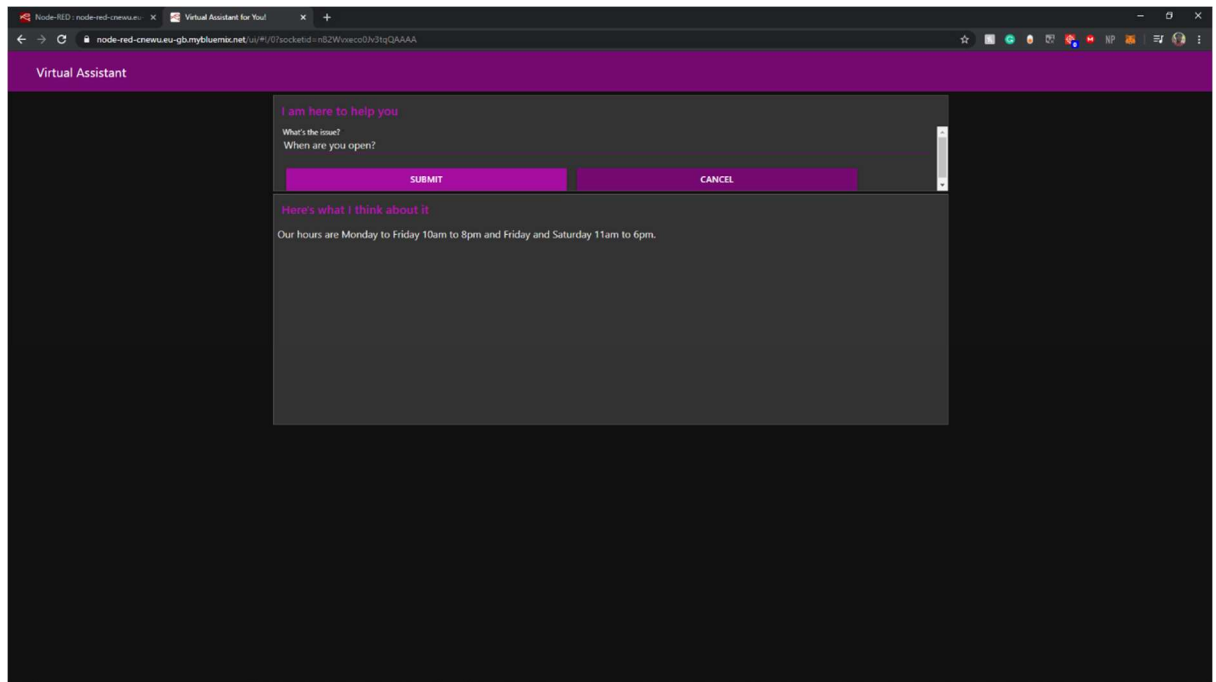


✓ This is the flowchart of the Node-Red

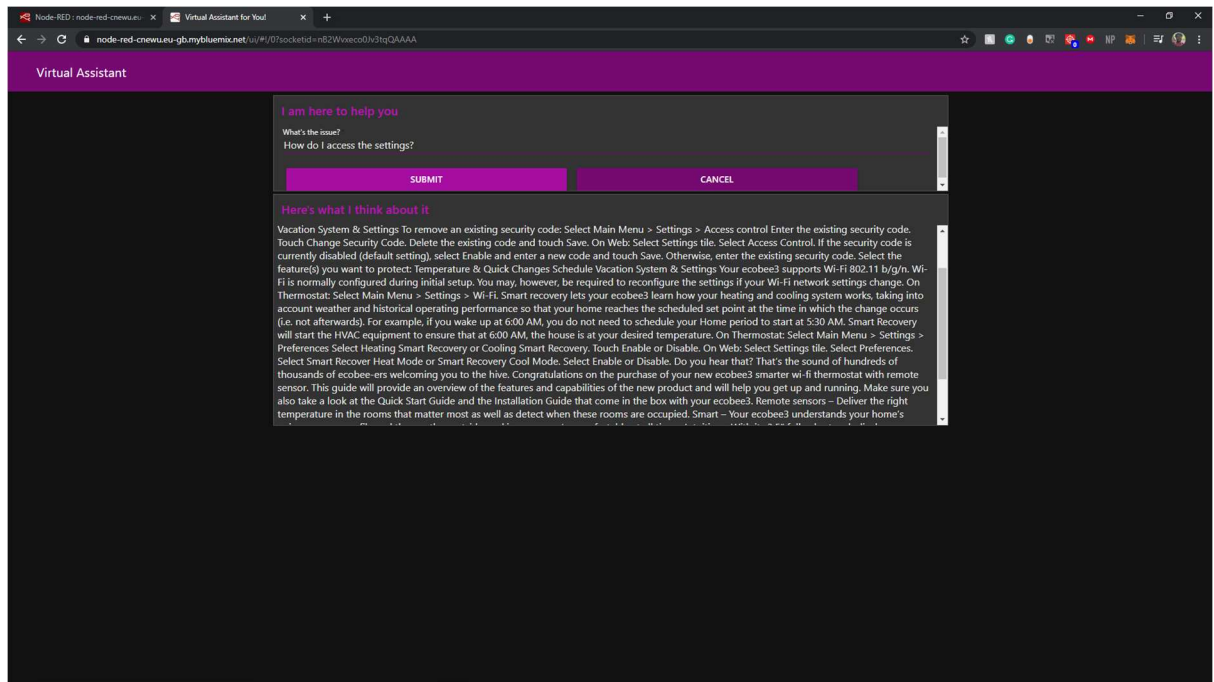
## 7. RESULTS

✓ When we click on the deploy, it get's deployed on this URL, "<https://node-red-crewu.eu-gb.mybluemix.net/ui/#!/0?socketid=SCyueWgzC7zPInNAAAAB>".









## 8. ADVANTAGES & DISADVANTAGES

### Advantages

- ✓ We can deploy this chatbot to decrease human interaction.
- ✓ Shares the workload.
- ✓ Cost Efficient.

### Disadvantages

- ✓ Chatbots can't always give the right solution
- ✓ Chatbots cannot feel the problem of the customer.

## 9. APPLICATIONS

It can be deployed on various platforms by integrating it with the platform you are using.

## 10. CONCLUSION

Congratulation, we have successfully created a chatbot with Smart Document Understanding,

using the various functionalities provided by IBM Cloud.

## 11. FUTURE SCOPE

We can include speech synthesis, and make it hands free.

## 12. BIBLIOGRAPHY

Source Code:

Cloud Functions:

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

```

## Node-Red Flow

```

[{"id":"91bfe3c2.82a68","type":"tab","label":"Customer Help Desk","disabled":false,"info":""},{
  "id":"cc38974d.042738","type":"debug","z":"91bfe3c2.82a68","name":"","active":true,
  "tosidebar":true,"console":false,"tostatus":false,"complete":false,"x":1130,"y":120,
  "wires":[ ]},{
  "id":"d09bb681.085fd8","type":"function","z":"91bfe3c2.82a68","name":"","func":
  "msg.payload = \" \" + msg.payload.i

```

```

input\return msg;","outputs":1,"noerr":0,"x":230,"y":340,"wires":[["ca0286d2.1
82658"]]],{"id":"86722136.1bab1","type":"function","z":"91bfe3c2.82a68","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+\"\\\"\\n\\\"+msg.payload.context.webhook_re
sult_1.results[i].text;\\n    }\\n    msg.payload = msg.payload.text;\\n}\\nelse\\n
msg.payload = msg.payload.output.text[0];\\nreturn msg;","outputs":1,"noerr":0,
"x":870,"y":480,"wires":[["bf72349d.bd0368"]]],{"id":"3bcef049.c80bd","type":"
ui_form","z":"91bfe3c2.82a68","name":"","label":"","group":"4dbda236.78d2ac","
order":2,"width":0,"height":0,"options":[{"label":"What's the issue?","value":
"input","type":"text","required":true,"rows":null},"formValue":{"input":""},"
payload":"","submit":"Submit","cancel":"Cancel","topic":"","x":110,"y":660,"wi
res":[["d09bb681.085fd8"]],"info":"<input type=\\\"text\\\" placeholder=\\\"What's y
our query?\\\" />"},"id":"ca0286d2.182658","type":"watson-conversation-
v1","z":"91bfe3c2.82a68","name":"","workspaceid":"6e972715-324f-45a7-979e-
d609831a3679","multiuser":false,"context":false,"empty-
payload":false,"service-endpoint":"https://api.eu-
gb.assistant.watson.cloud.ibm.com/instances/fe1e54ee-a2dc-4084-9a58-
717f3aeb35ca","timeout":"","optout-
learning":false,"x":580,"y":120,"wires":[["cc38974d.042738","86722136.1bab1"]
]],{"id":"bf72349d.bd0368","type":"ui_template","z":"91bfe3c2.82a68","group":"3
69f99d6.04ccc6","name":"ChatBot","order":0,"width":"20","height":"6","format":
"<div ng-bind-
html=\\\"msg.payload\\\">\\n</div>","storeOutMessages":true,"fwdInMessages":true,"r
esendOnRefresh":true,"templateScope":"local","x":1200,"y":640,"wires":[[]]},{
" id":"4dbda236.78d2ac","type":"ui_group","z":"","name":"I am here to help you",
"tab":"5ccf1450.7d554c","order":1,"disp":true,"width":"20","collapse":false,"i
nfo":"<input type=\\\"text\\\" placeholder=\\\"Enter your problem here\\\" />"},"id":
"369f99d6.04ccc6","type":"ui_group","z":"","name":"Here's what I think about i
t","tab":"5ccf1450.7d554c","order":2,"disp":true,"width":"20","collapse":false
},{ "id":"5ccf1450.7d554c","type":"ui_tab","z":"","name":"Virtual Assistant","i
con":"dashboard","disabled":false,"hidden":false}]

```

Reference:

1.

<https://www.ibm.com/cloud/architecture/tutorials/cognitive-discovery> 2.

<https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started>

3. [https://developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chat bot-on-nodered/](https://developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chat-bot-on-node-red/)

4. <http://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red>

5. <https://github.com/IBM/watson-discovery-sdu-with-assistant>