Project Report

Project Topic **: -** Intelligent Customer Help Desk with Smart Document Understanding.

Category **: -** Machine learning Engineer

Name **: -** Shivendra Patel

Email ID : - Shivendra123patel2gmail.com

Internship **: -** Smart Bridge (Remote Summer Internship Program 2020)

Website **: -** https://smartinternz.com/

CONTENTS

1. Introduction

1.1 Overview

In this project we will be able to write an application with the help of IBM Cloud Services (Watson AI services like Discovery, Assistant, Cloud Function and Node Red). By the end of the project we will get a Chatbot which act as an intelligent web based Customer support Chatbot or after sales Service and Customer queries.

• Project Requirements: IBM Cloud account, Python, Node Red, IBM Watson Services.

• Functional Requirements: IBM Cloud Services

• Technical Requirements: Python, Node JS, Watson AI, ML

• Software Requirements: Watson Assistant, Watson Discovery, IBM Cloud Function, Node Red

• Project Deliverables: Smartinternz Internship

• Project Duration: 1 Month

1.2 Purpose

When we purchase any electronic machine then a customer care will explain everything about the electronic goods. Basically Chatbot is just like a customer care which usually can answer just simple questions, such as store locations and hours, directions and maybe even making appointments. When any user input (question) falls out of the scope of the predetermined question set of Chatbot, it typically tells us to rephrase our sentence or mention that this question is not valid.

In this project, there is another option. If the customers' question is about the operat ion of a device, the application passes the question onto the IBM Watson Discovery Service,which has been preloaded with the device's owners manual. So now, instead of "Wouldyou like to speak to a customer representative?" we return relevant sectio ns of the owner are manual to help solve our customers' problems.
To take it a step further, the project uses the Smart Document Understanding featur of IBM Watson Discovery Service to train it on what text in the owner's manual is Important and what is not. This improves the answers returned from the queries.

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 po st queries to Watson Discovery.
- Build a web application with integration to all these services and deploy the s ame on IBM Cloud Platform.

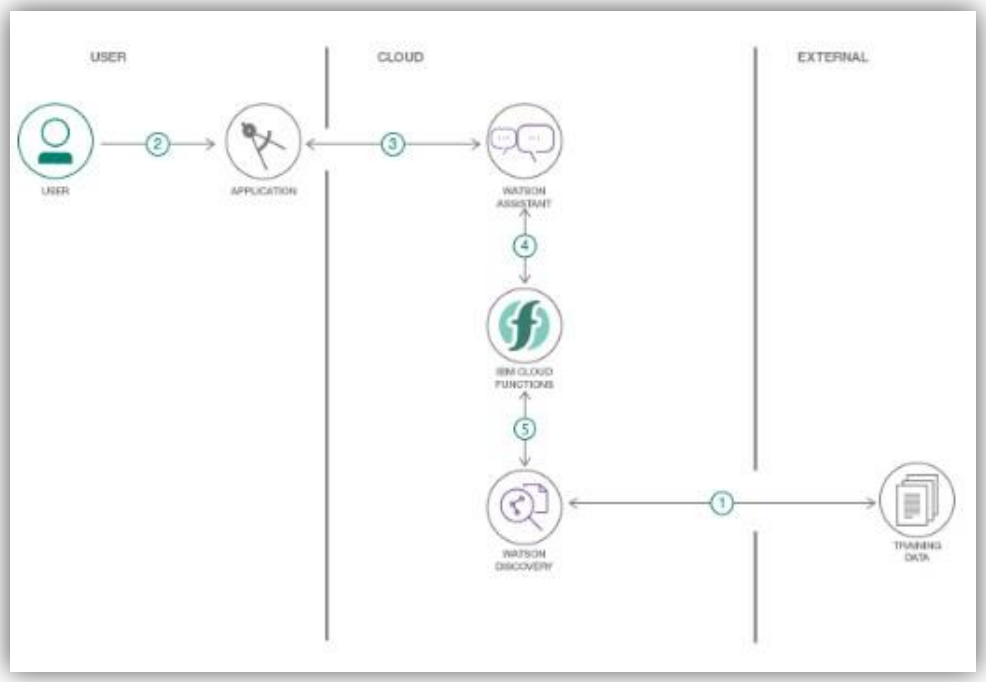
2.1 Existing Problem

Generally Chatbot are loaded with a certain set of questions that is more like if and else flow, the question or the user input which lies out of the scope of the chatbot is not answered and rather a message like “Try again after rephrasing” & “I am unable to understand, Please Rephrase” are displayed and it directs the user to the customer agent or the representative but an efficient chatbot should reduce the traffic reaching to the representatives, So to achieve this we include a Smart chatbot so that it can answer the queries of the customer.

2.2 Proposed Solution

In order to solve the above-mentioned problem, we have to put a virtual agent in chatbot so it can understand the queries that are posted by customers. The virtual agent should be trained from some insight based on company backgrounds, working hours, store locations and product related information. In this project I used Watson Discovery to achieve the above solution.

3.1. Block Diagram



- The document is annotated using Watson Discovery SDU.
- 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.
- Dialog between the user and backend server is coordinated using a Watson Assistant dialog skill.
- If the user asks the product operation question, a search query is passed to a predefined IBM Cloud Function action.
- Cloud function action will query the Watson Discovery service and return the results.

3.2. Hardware/Software Designing

- Create necessary IBM Cloud services.
- Configure IBM Watson Discovery
- Create IBM Cloud Function action
- Configure IBM Watson Assistant
- Create Node-RED flow and configure node
- Deploy and run node red app.

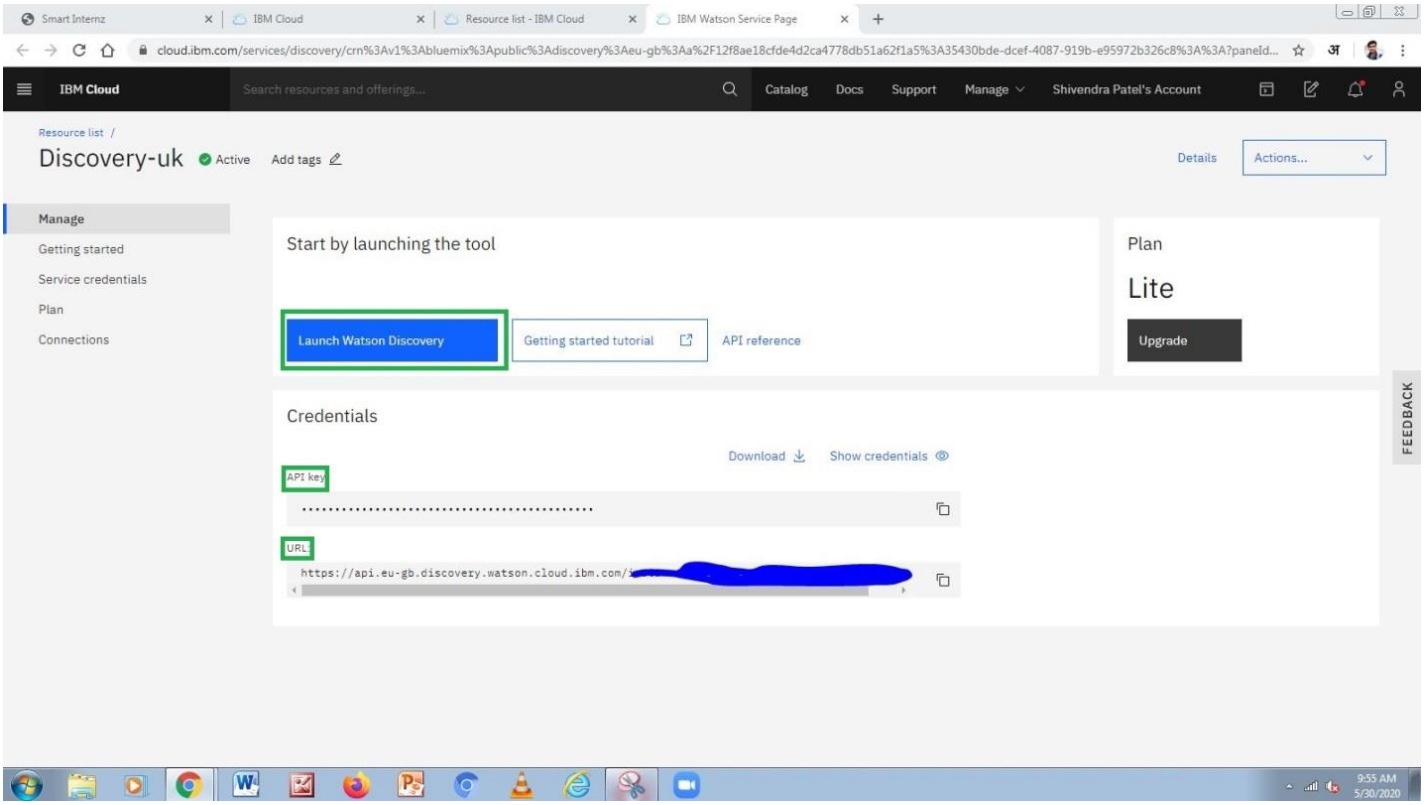
4. Experimental Investigation

1. Create required IBM Cloud Services

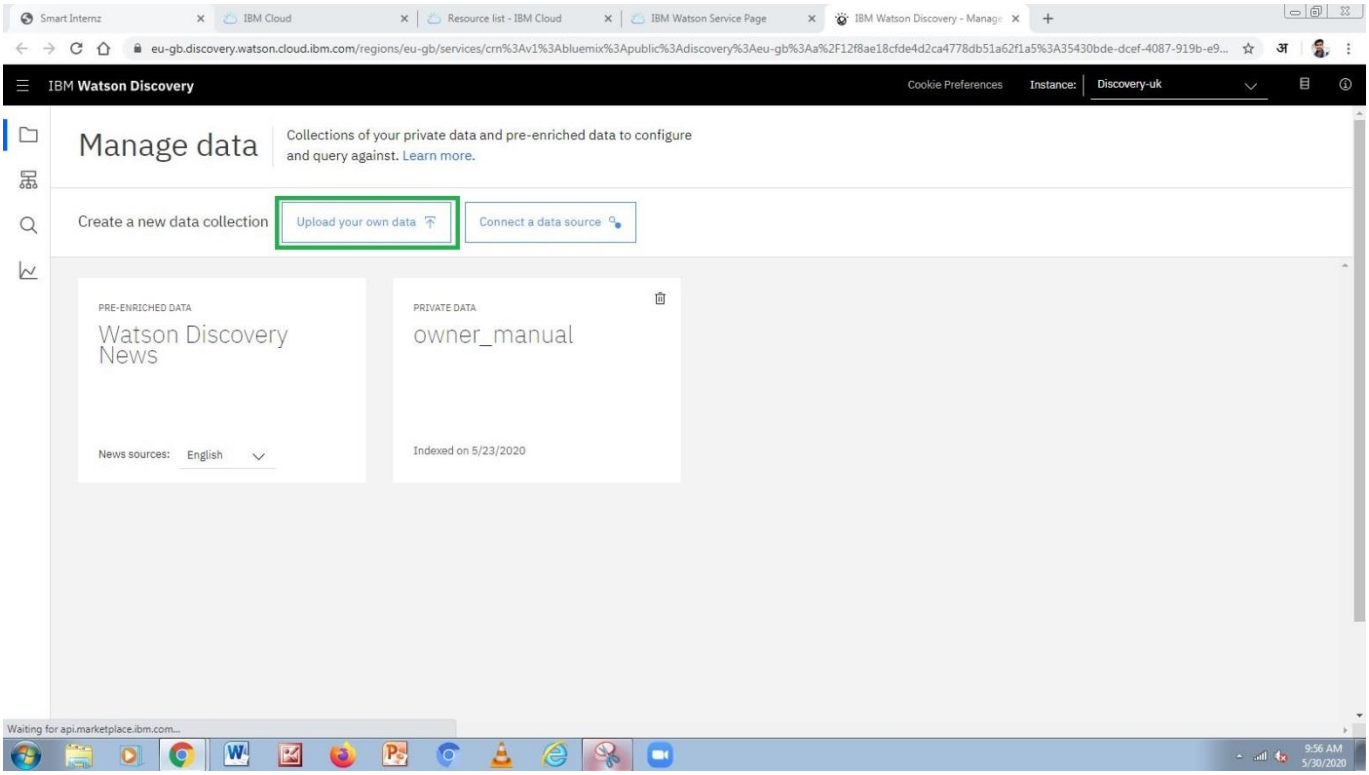
- IBM Watson Assistant
- IBM Watson Discovery
- Node-RED

2. Configure IBM Watson Discovery

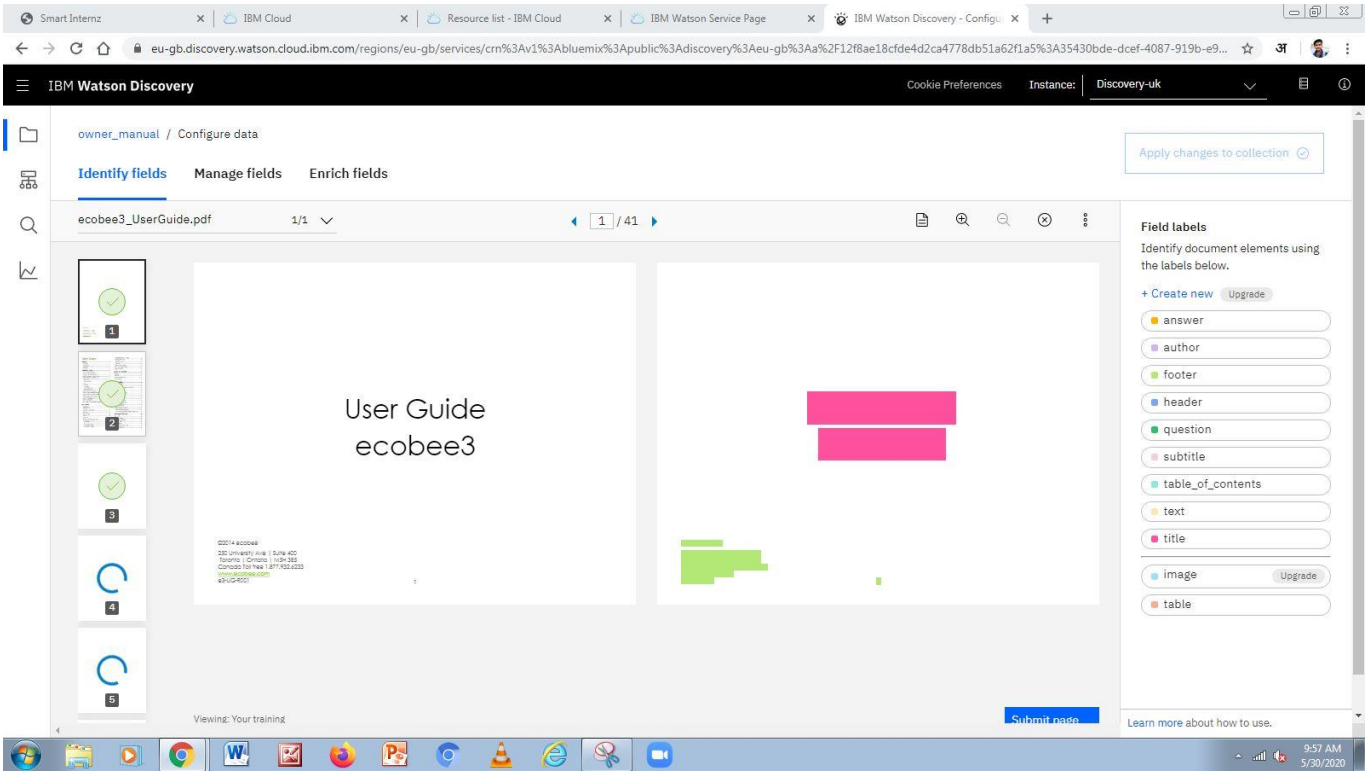
a. Launch Watson Discovery service.



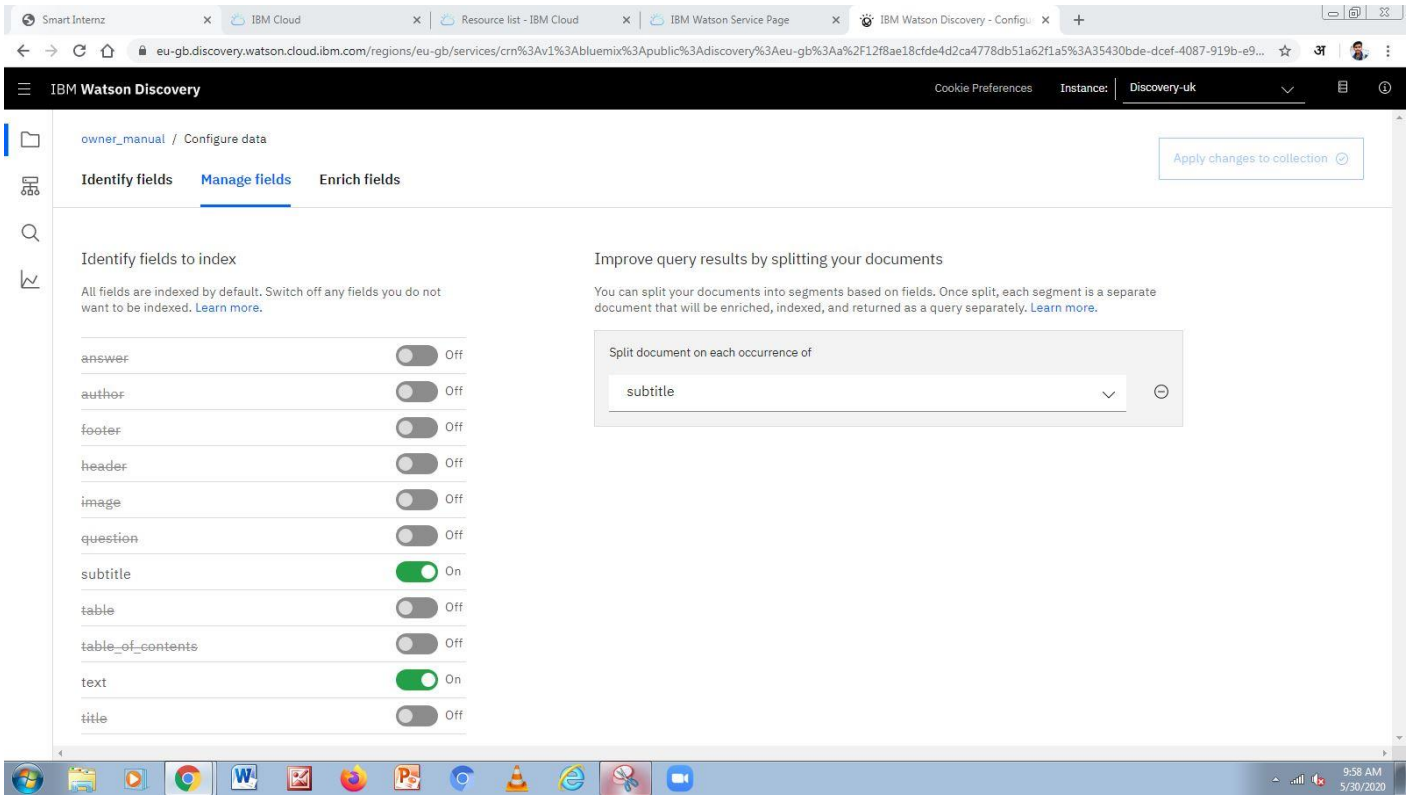
b. Upload documents for Smart Document Understanding (I have used the Ecobee3 Smart Thermostat User Manual).



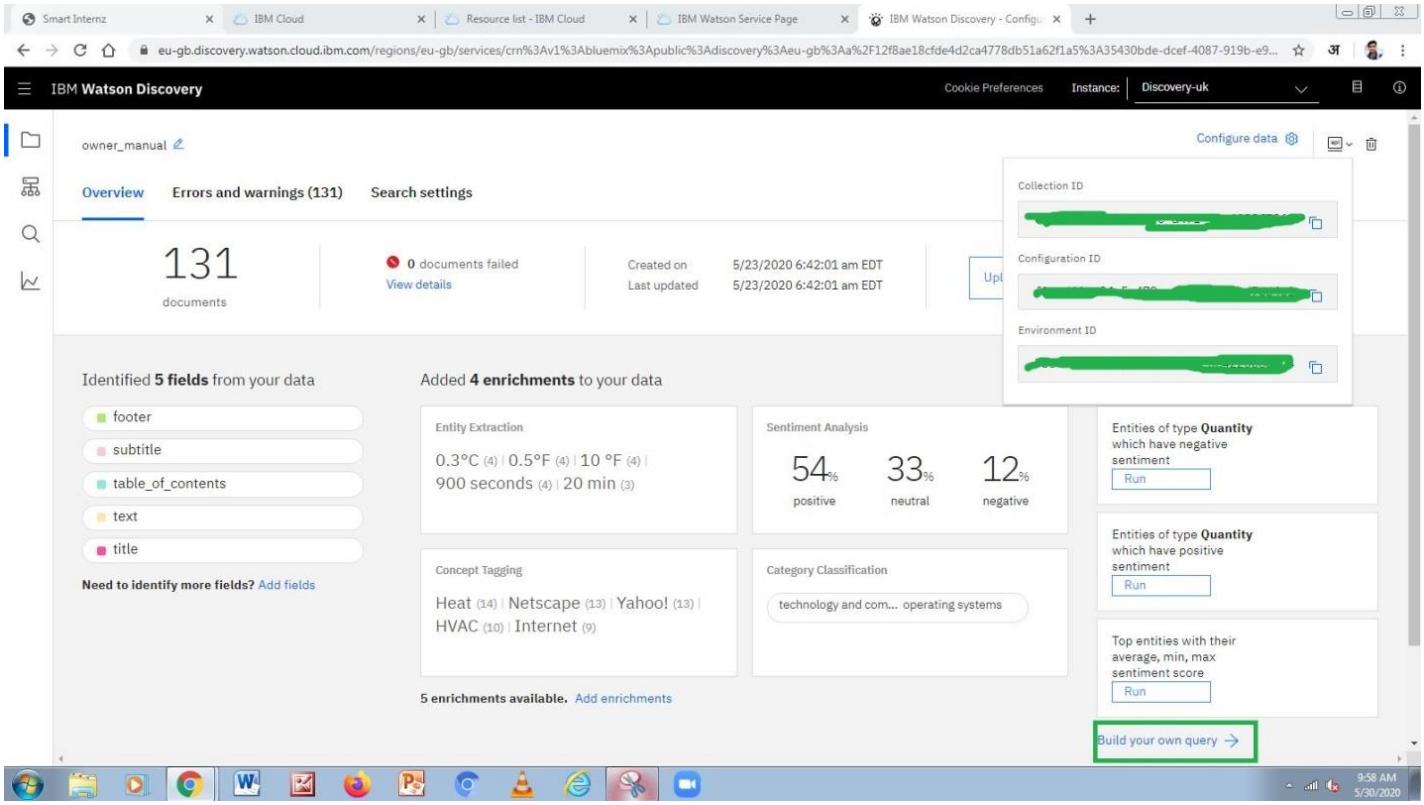
c. Annotate data for Smart Document Understanding.



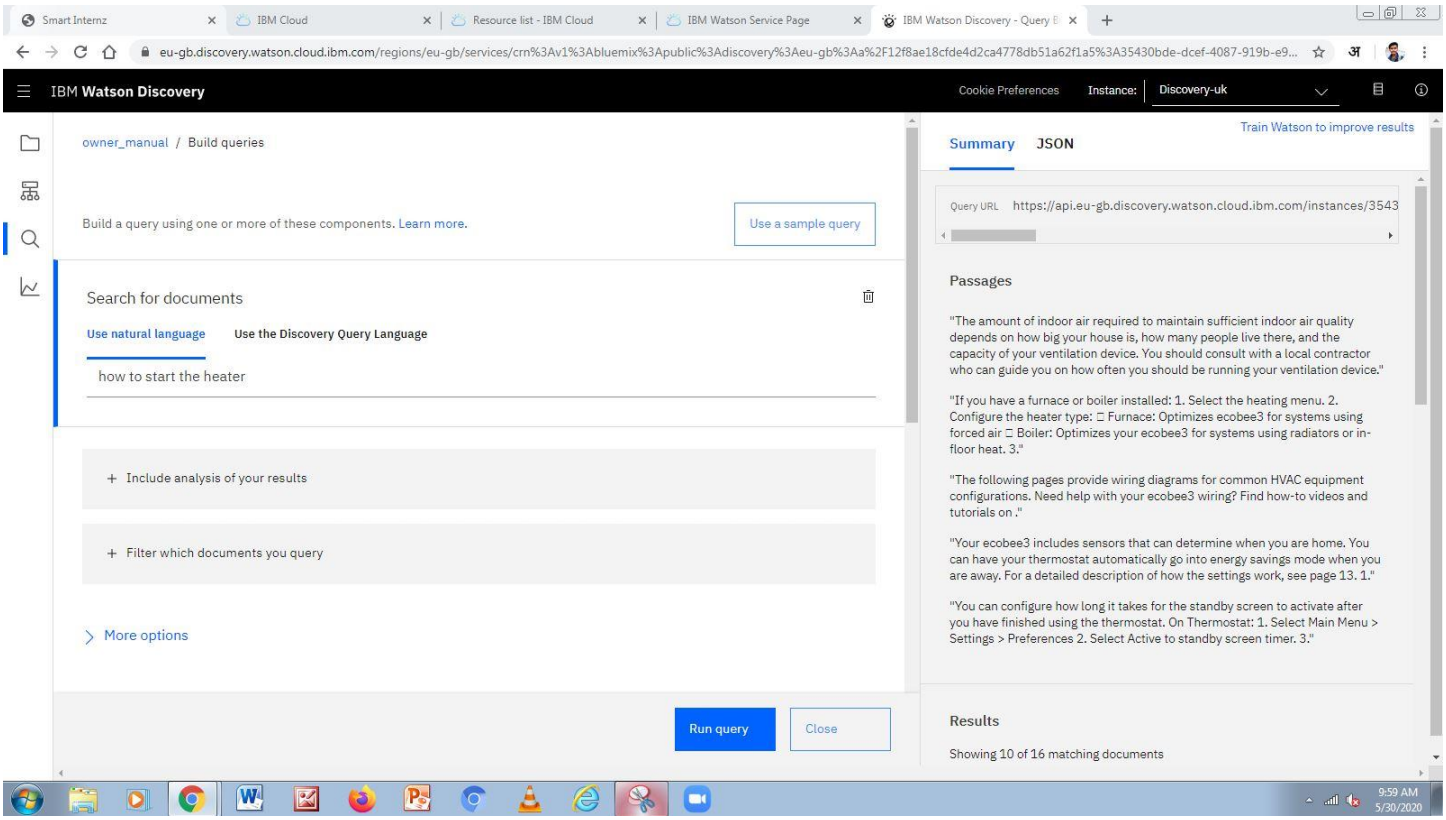
d. Select the fields to be indexed and split document for improved results (I have selected only subtitles and text for indexing and split the document using subtitles).



e. Run a query to check accuracy of discovery, store Discovery credentials and also the API key and URL shown step 2(a).

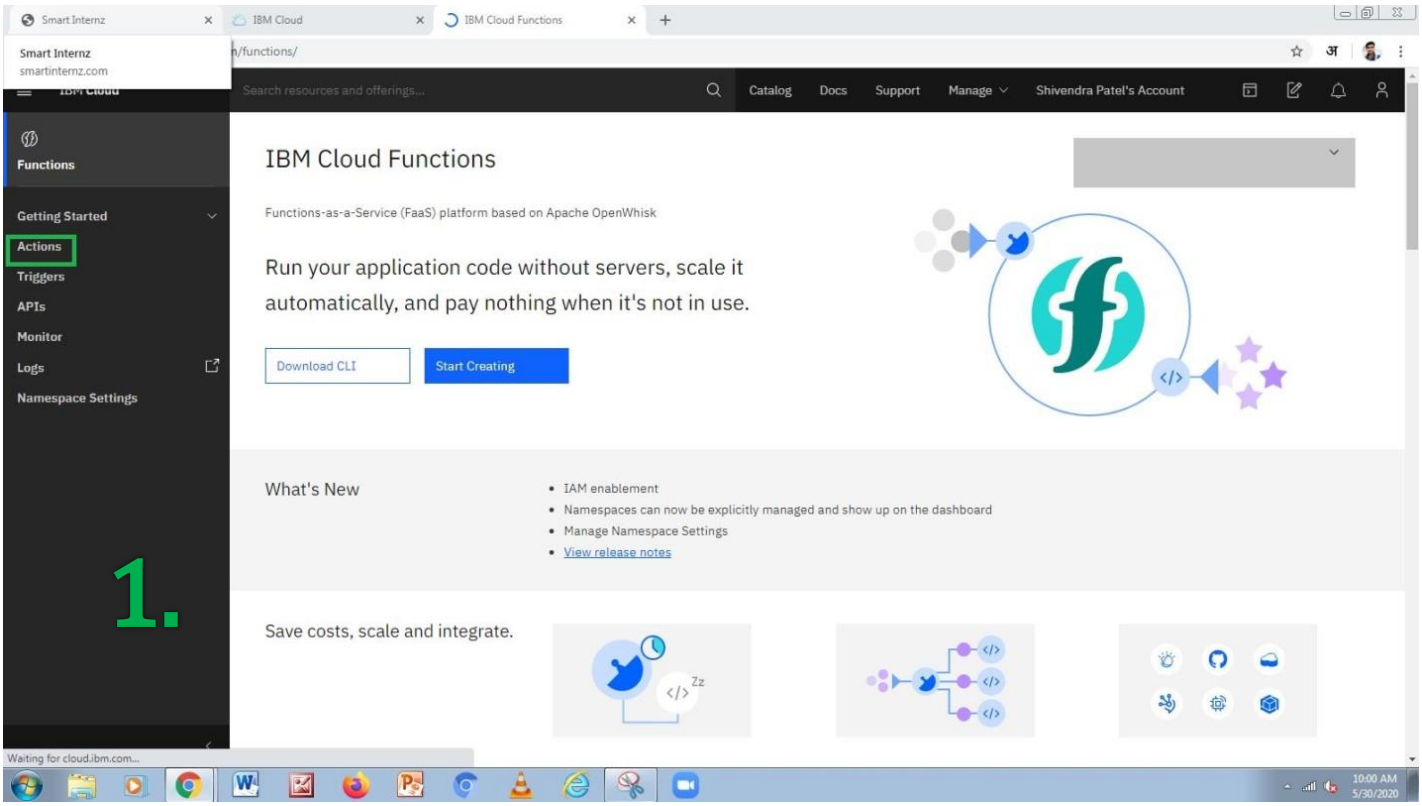


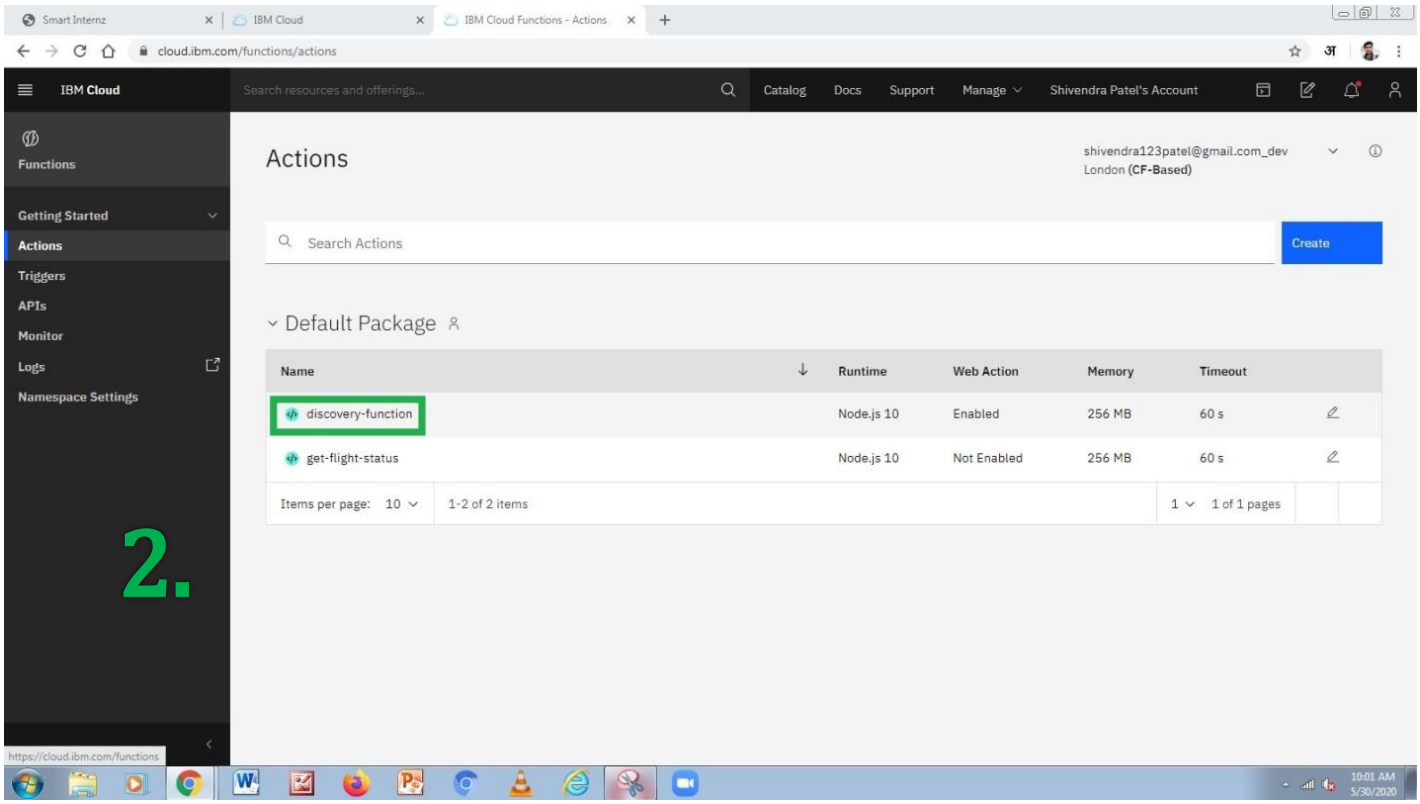
And then run your query.



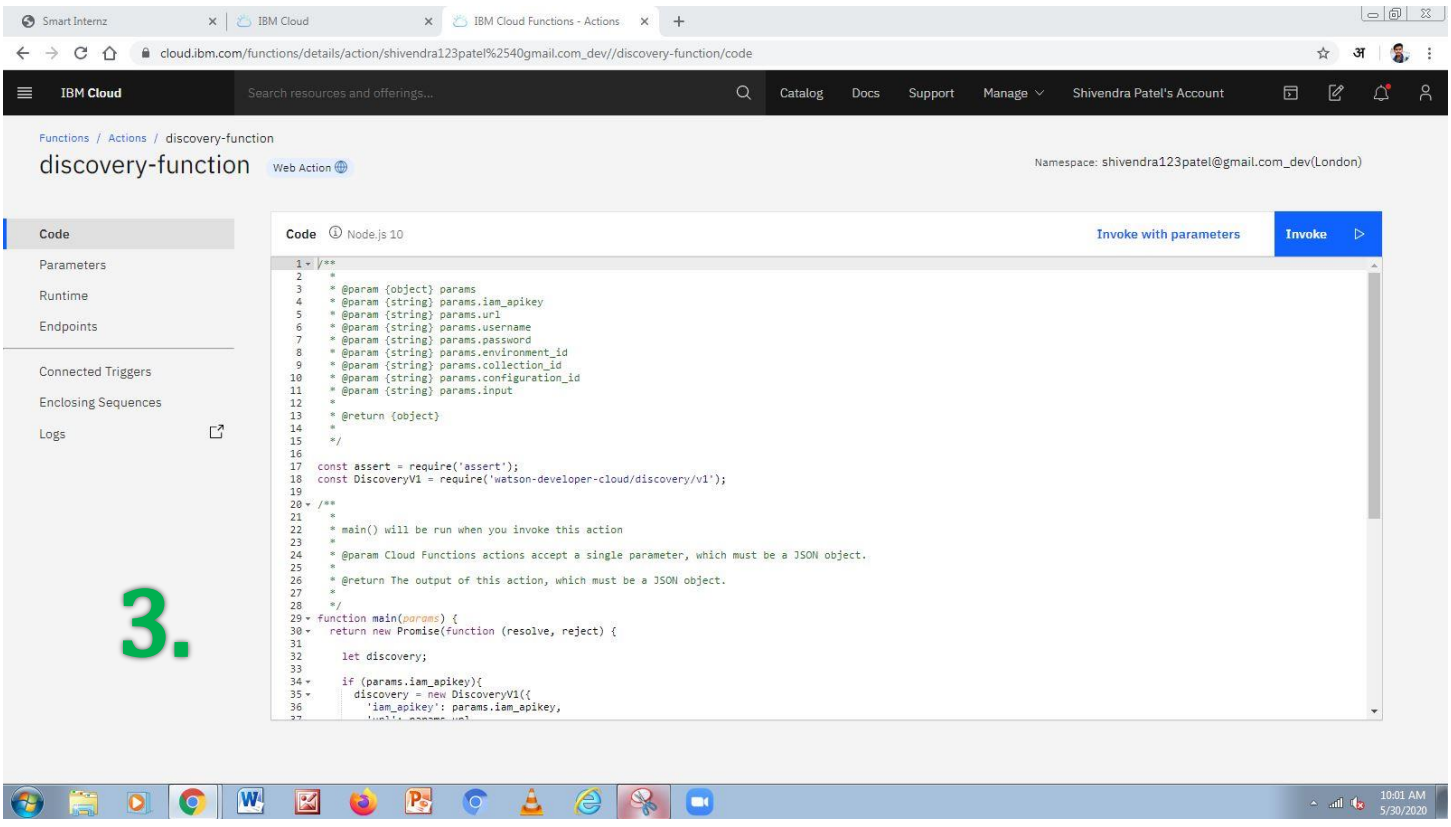
3. Create a webhook using IBM Cloud Functions Action

a) This is used to integrate IBM Discovery Service with IBM Watson Assistant using a webhook.

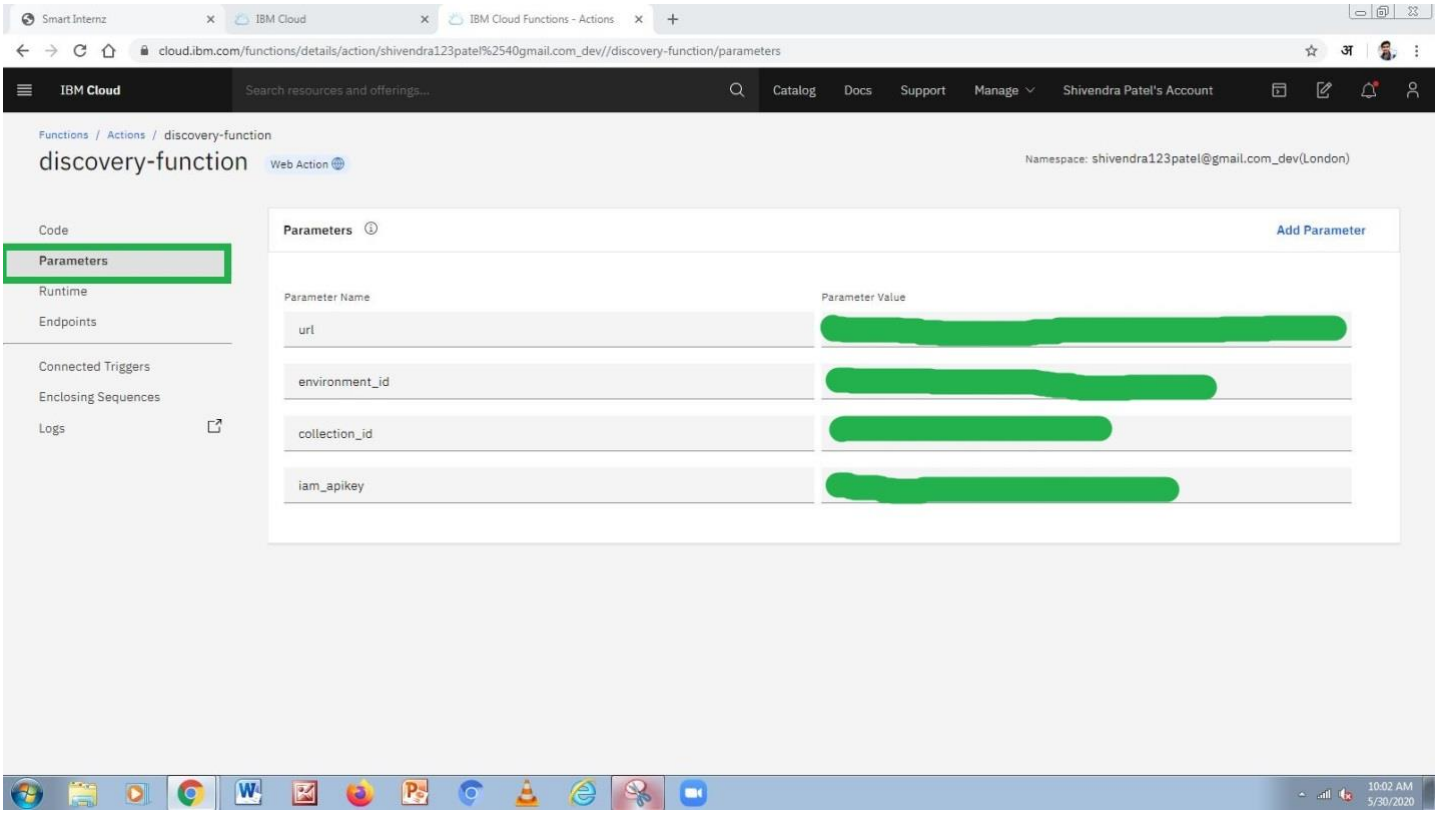




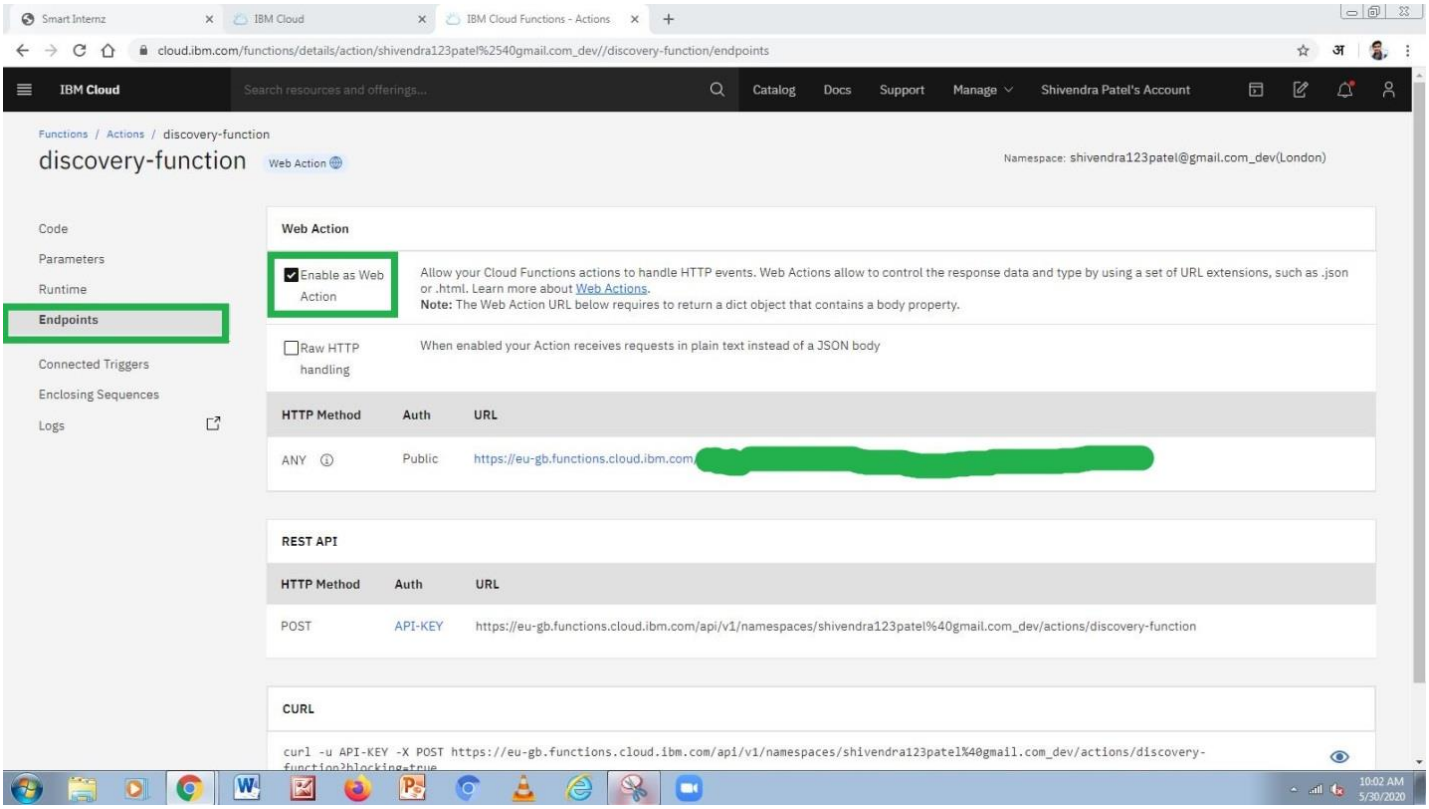
b. Create an IBM Cloud Functions Action (Source code is given in appendix).



c. Click on parameters, create required parameters and use the Discovery credentials for parameter values.

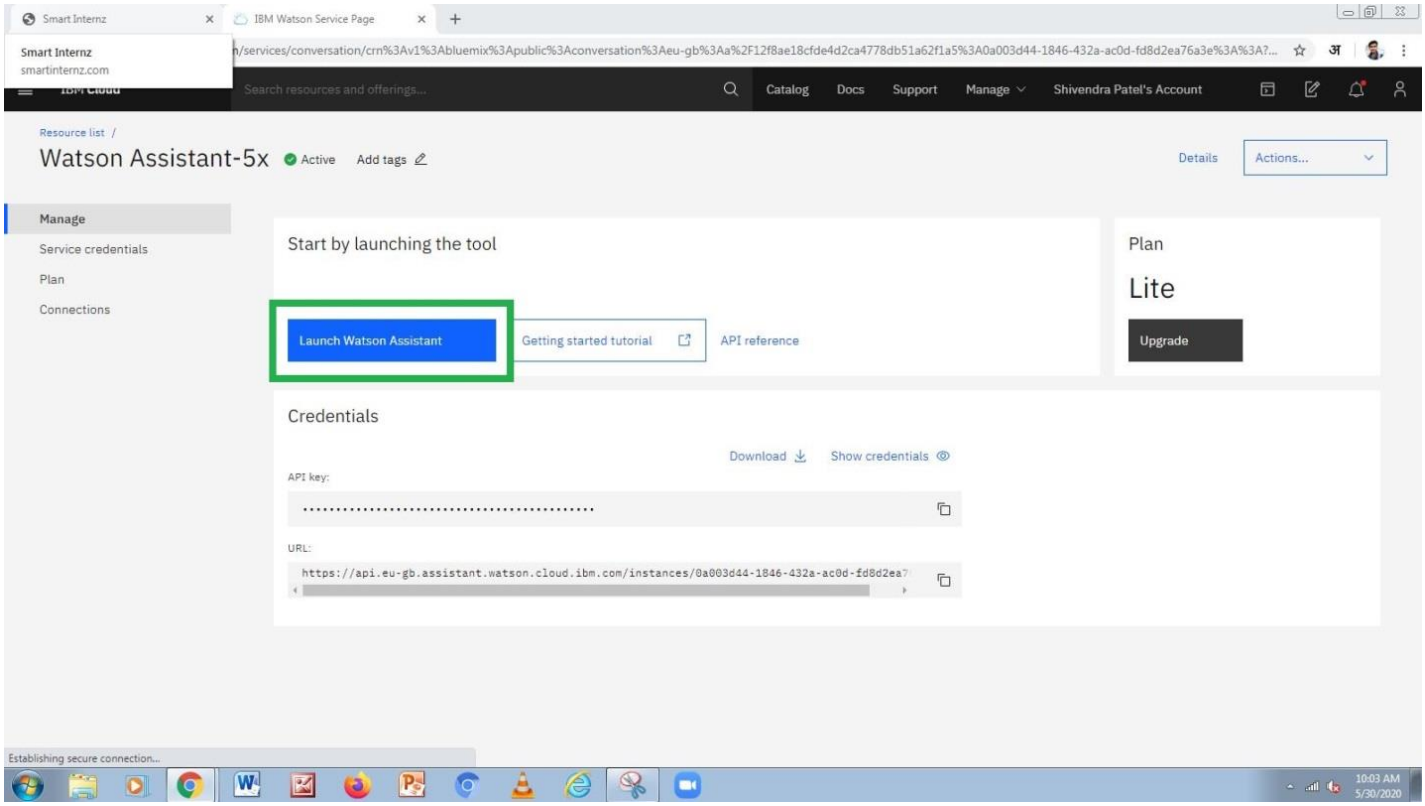


d. Go to Endpoints tab and click on the Enable as web action checkbox and store the URL.

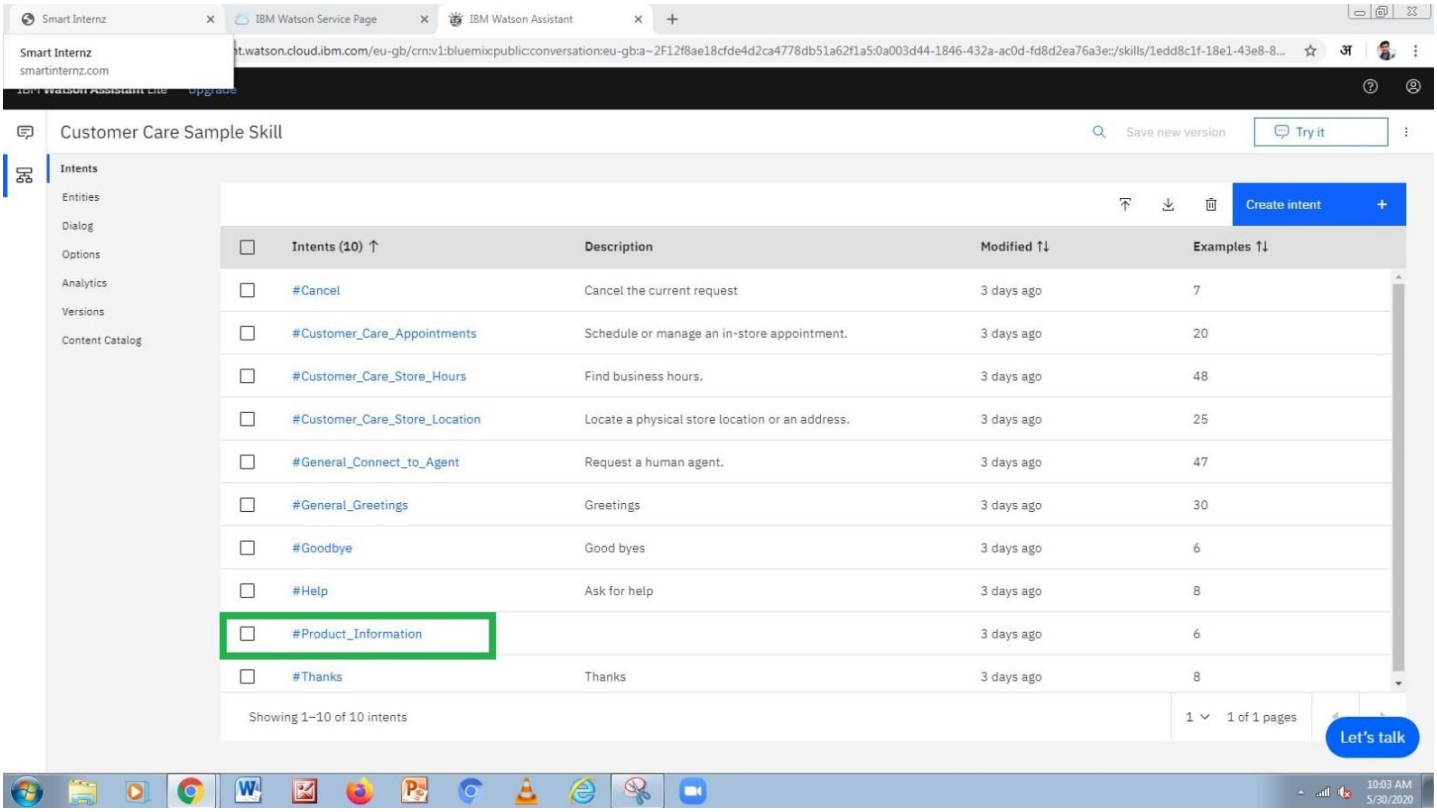


4. Configure IBM Watson Assistant

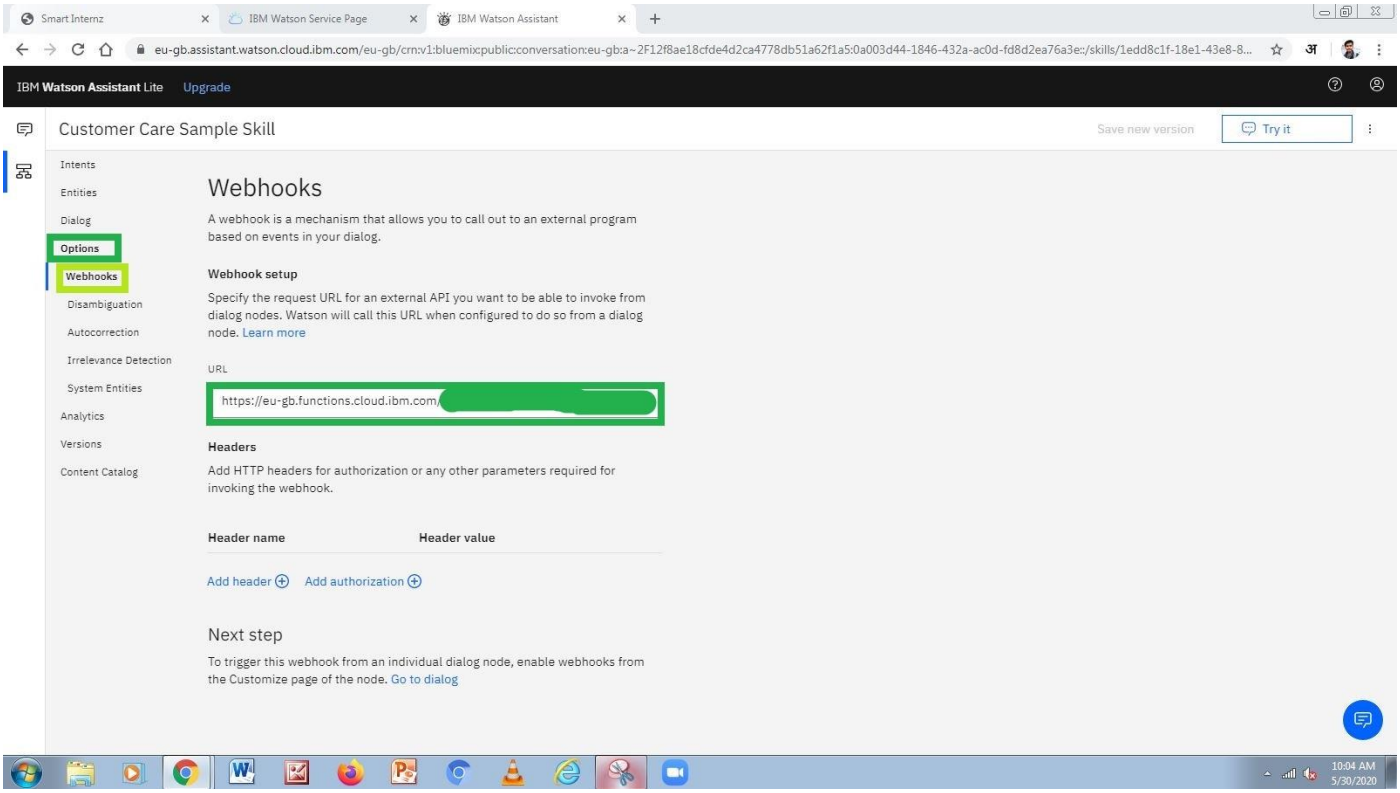
a. Launch IBM Watson Assistant and use the Customer Care Sample Skill to create an assistant with pre-loaded queries related to customer support.



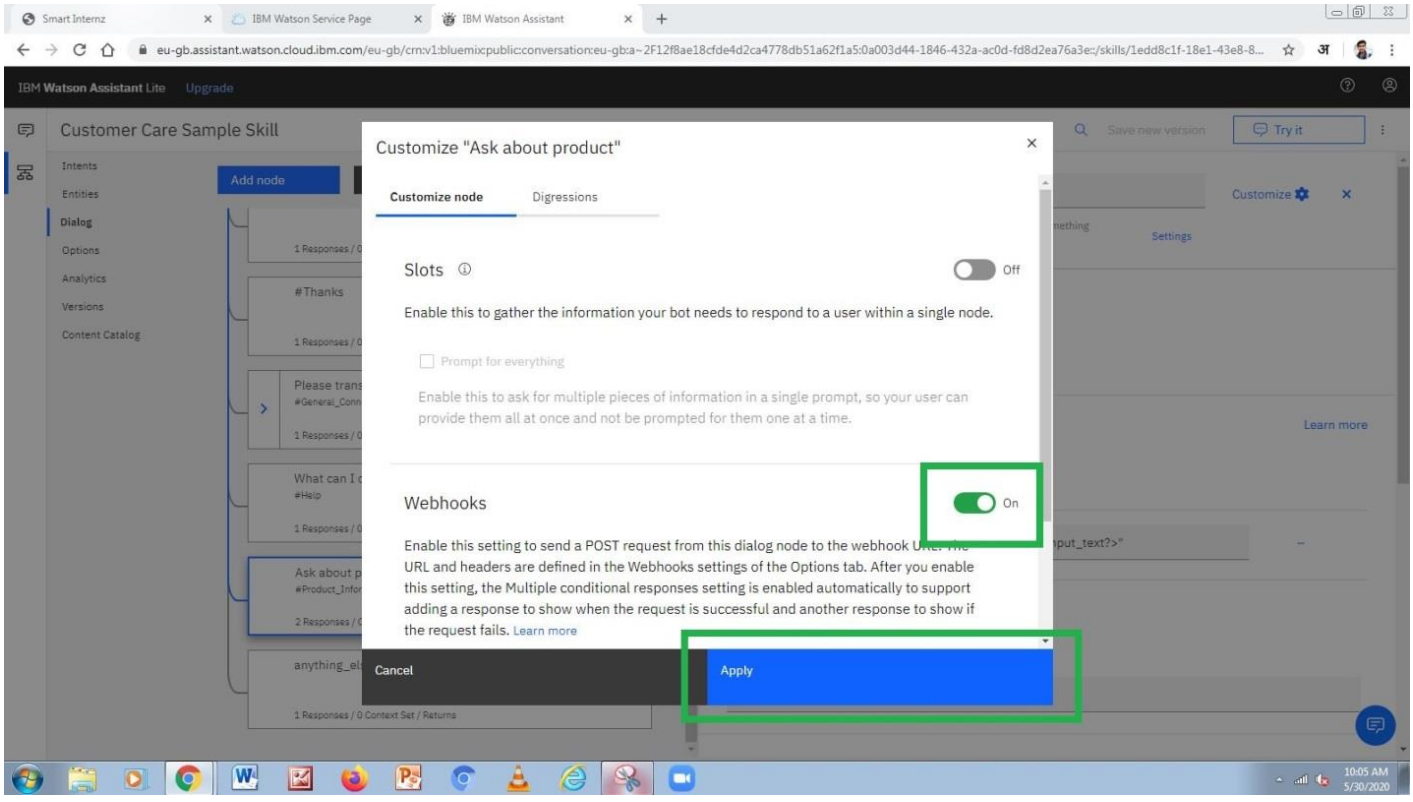
b. Create an intent for product related queries (I have created the Product Info intent) and add to dialog.



c. Go to options tab and under webhooks, paste the URL we stored in step 3(d).

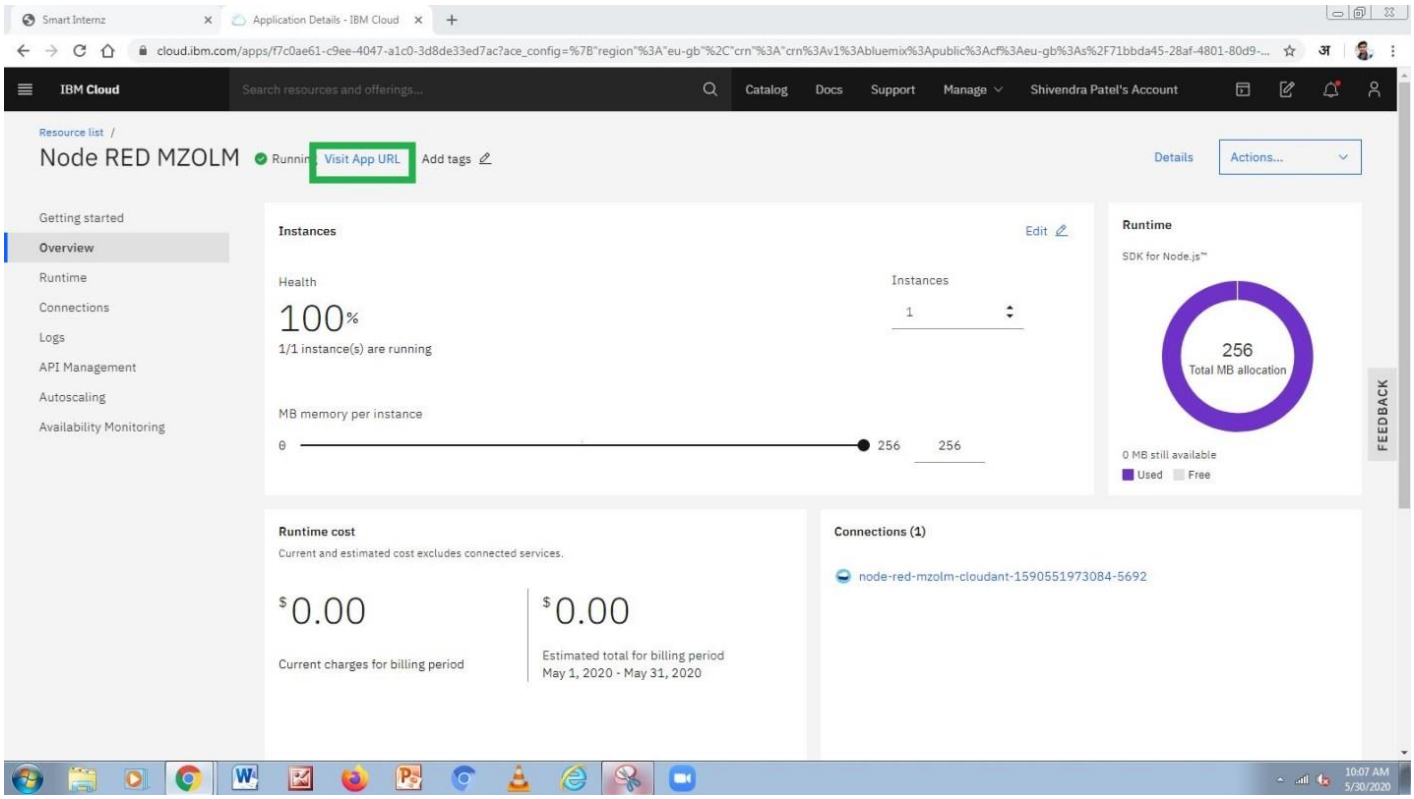


d. In dialog tab enable webhooks.

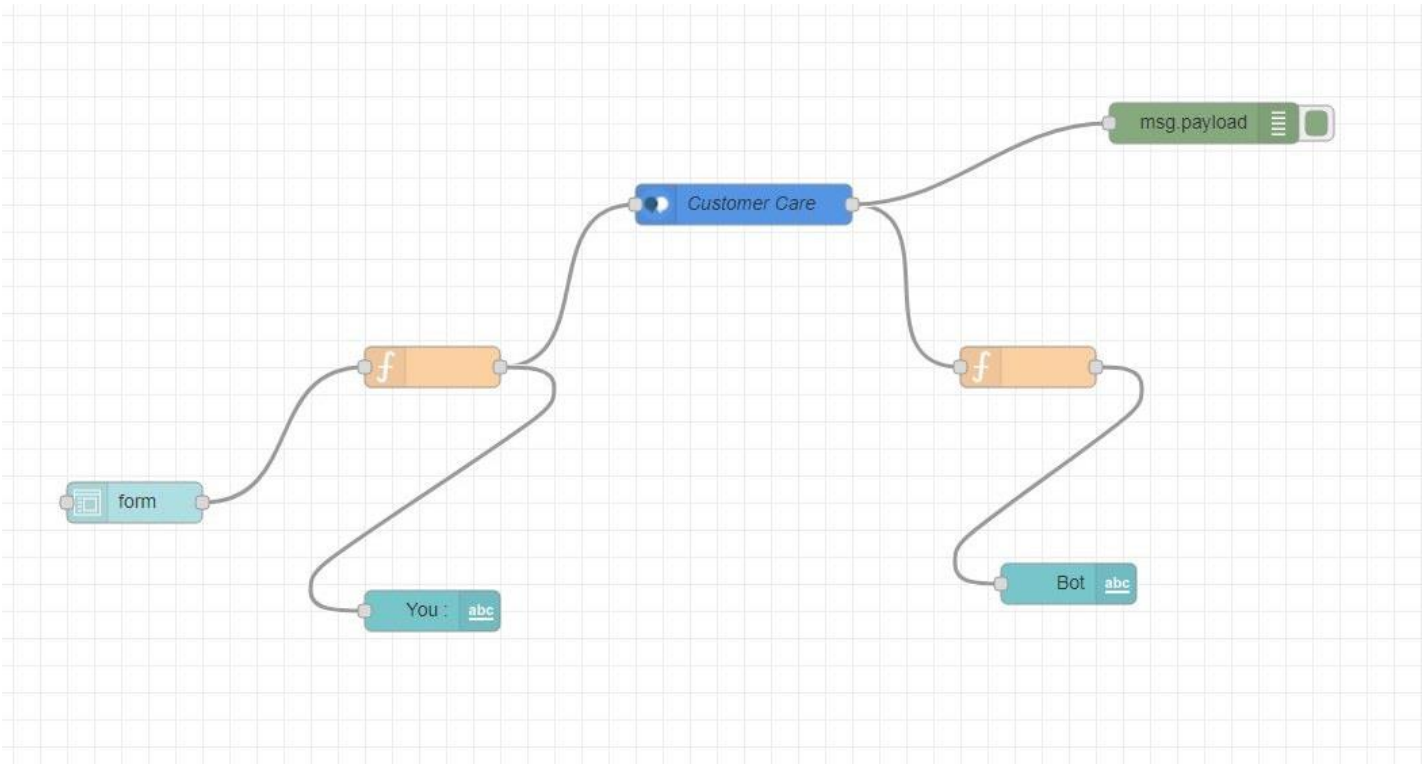


5. Create a Node-RED flow

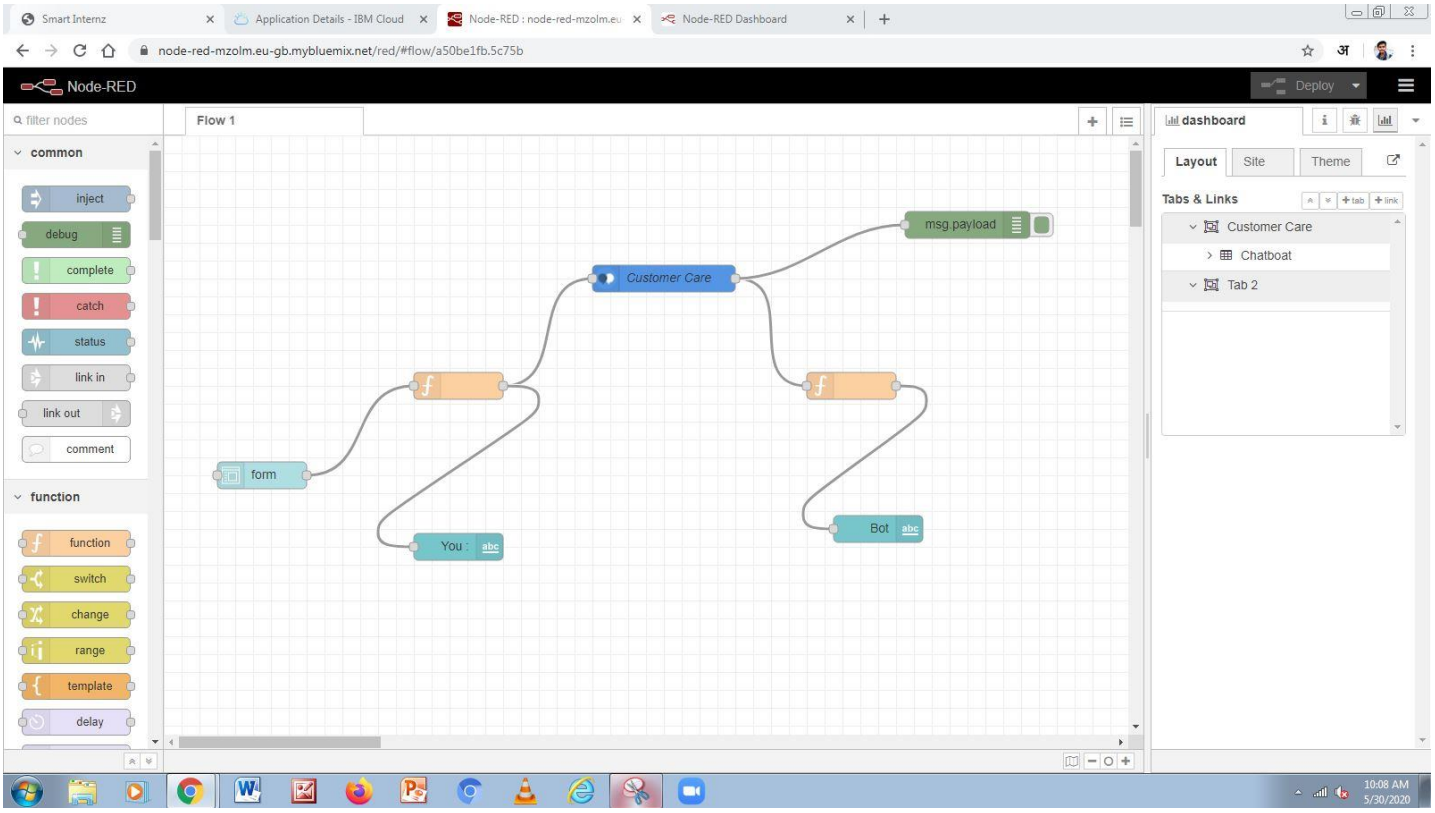
a. open Node Red app from the Cloud foundry apps.



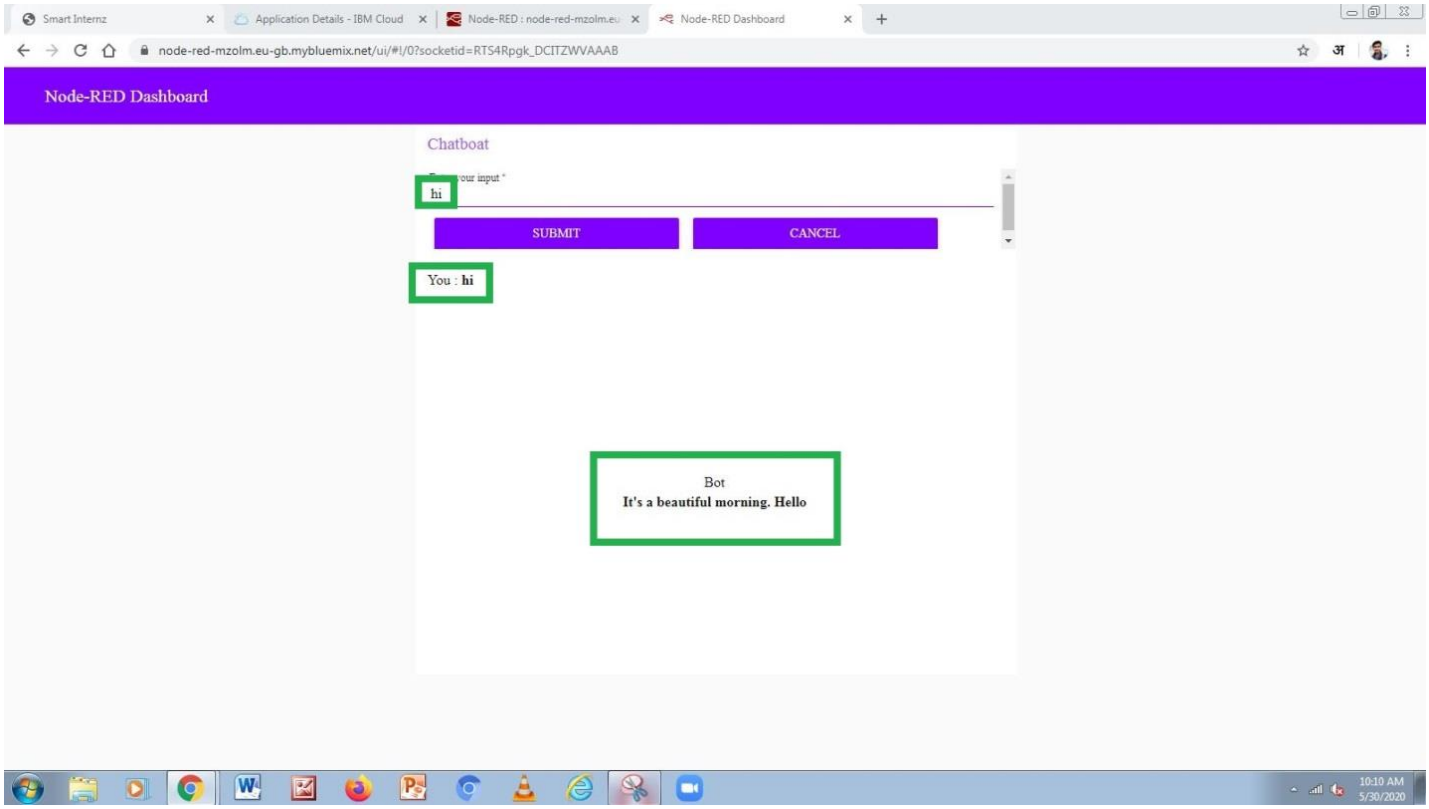
b. Use Node-RED to create a UI and to link all the services.

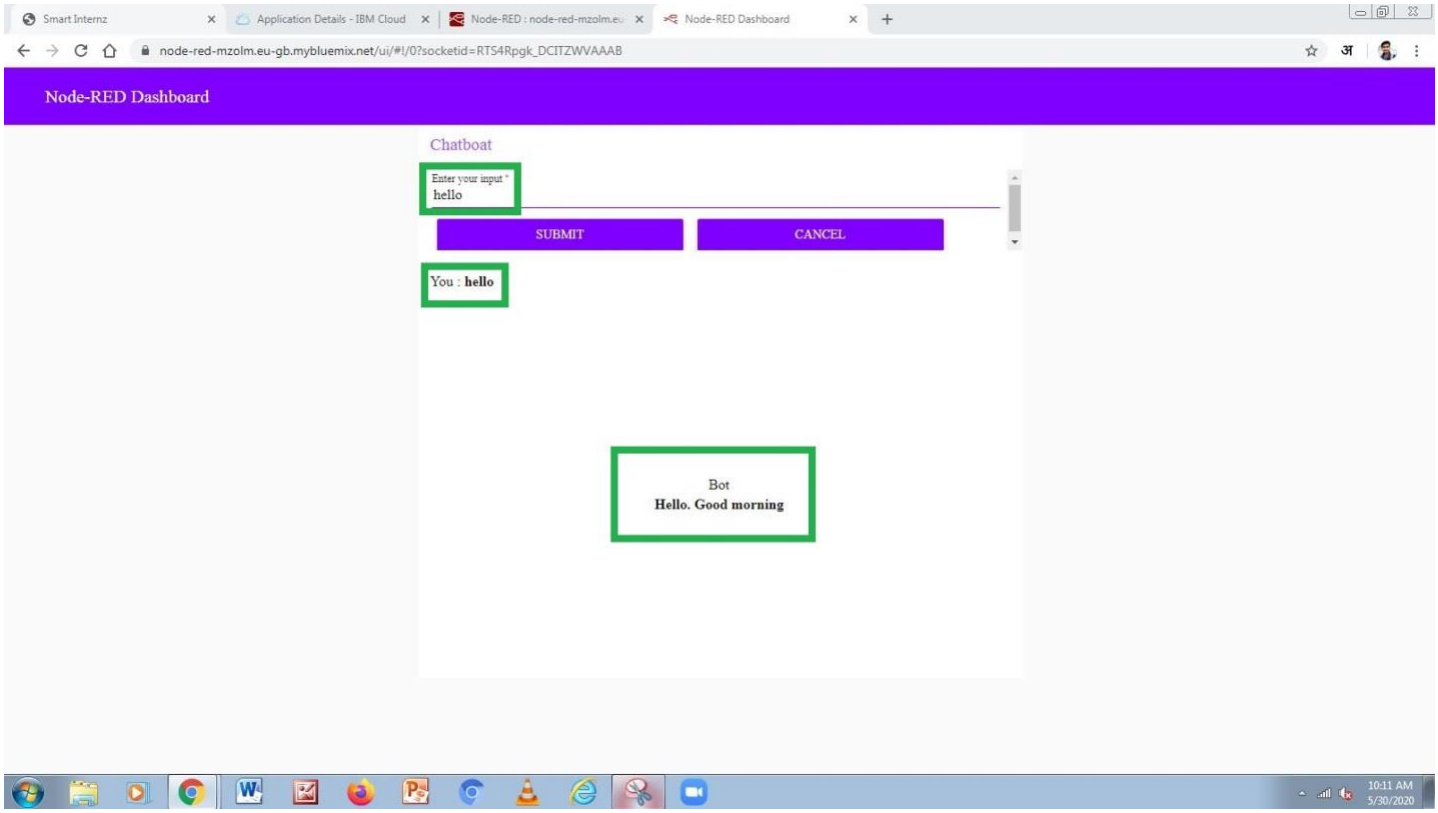


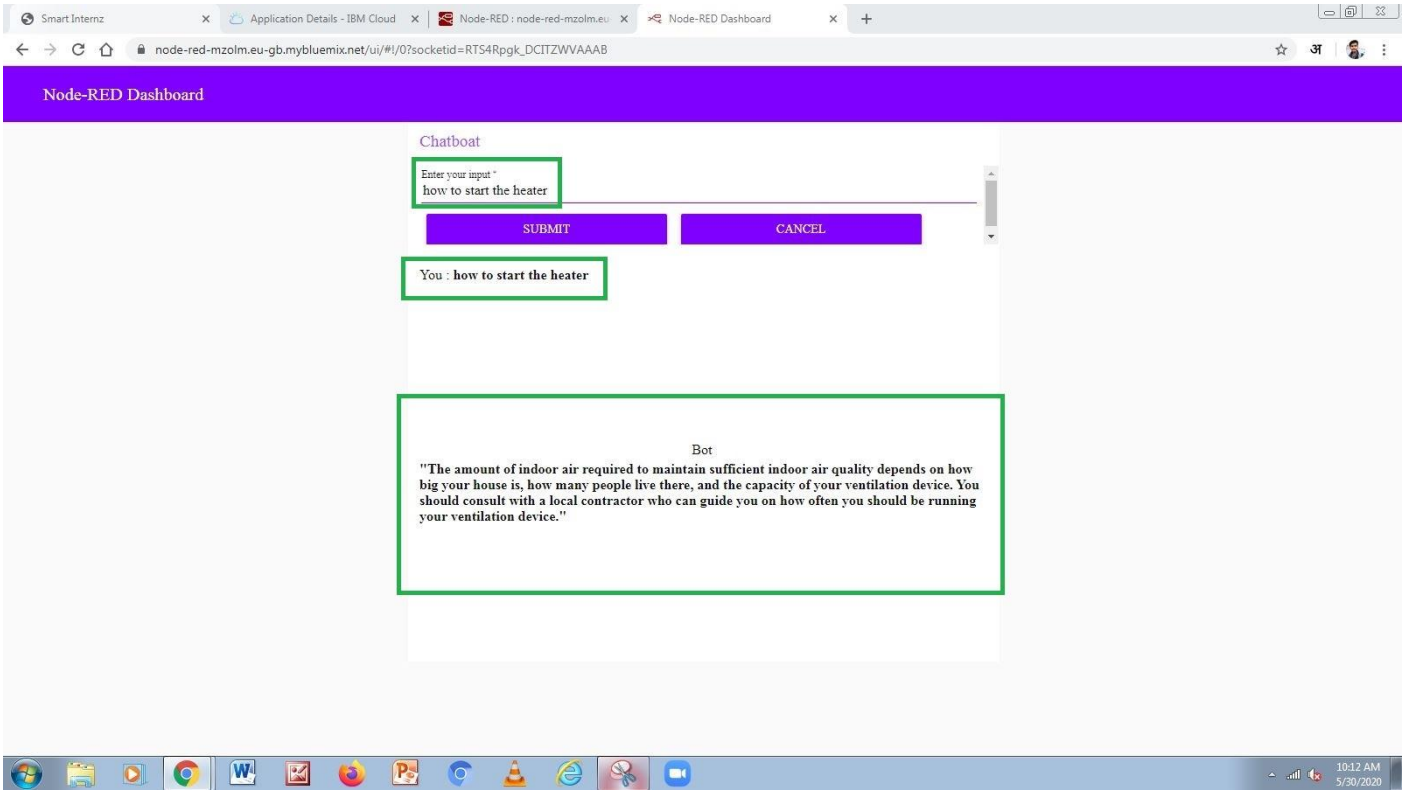
5. Flowchart

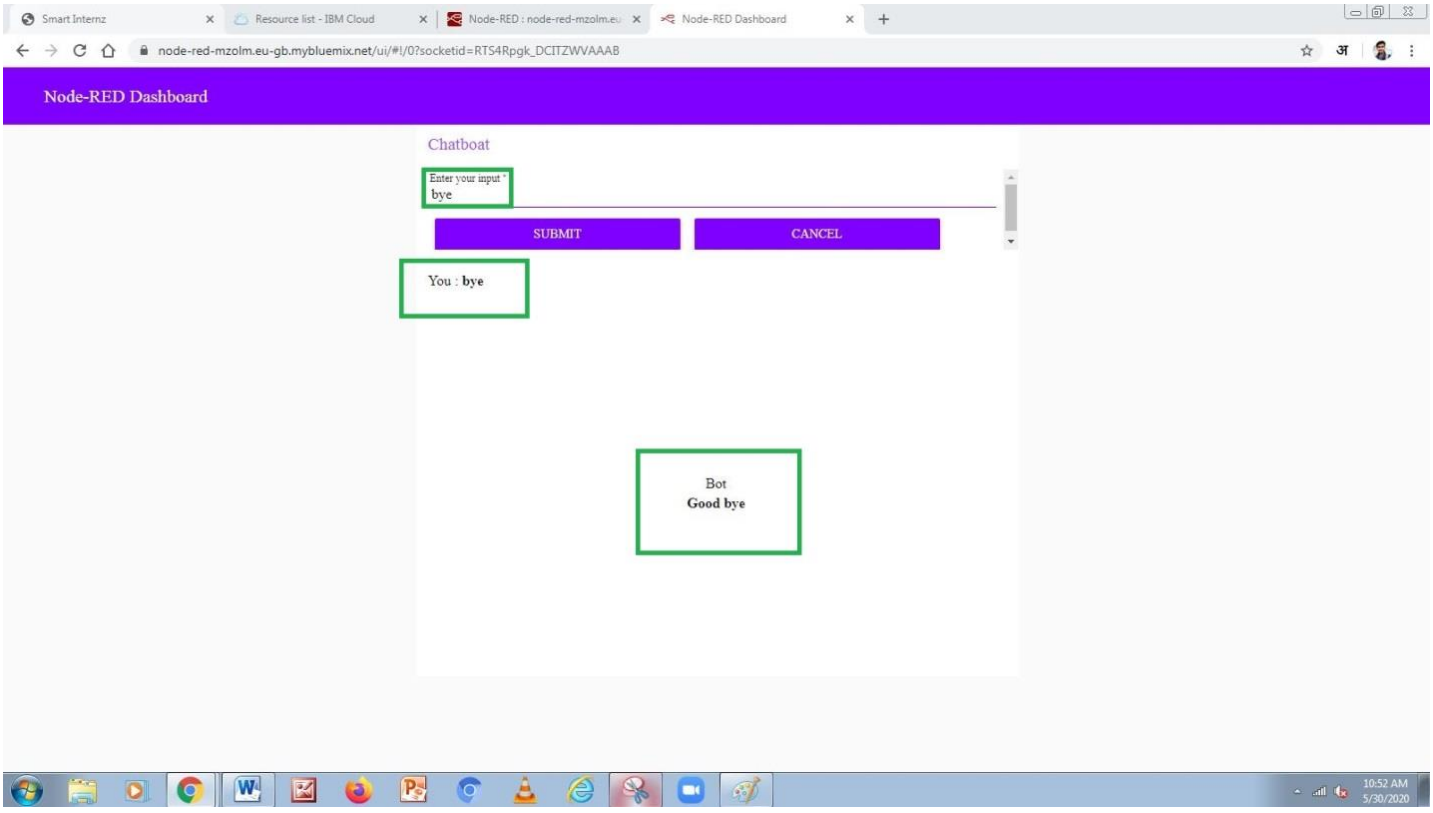

Smart Internz
Application Details - IBM Cloud
Node-RED : node-red-mzolm.eu
Node-RED Dashboard
node-red-mzolm.eu-gb.mybluemix.net/red/#flow/a50be1fb.5c75b
Node-RED
Deploy
filter nodes
Flow 1
common
inject
debug
complete
catch
status
link in
link out
comment
function
function
switch
change
range
template
delay
dashboard
Layout
Site
Theme
Tabs & Links
Customer Care
Chatboat
Tab 2
msg.payload
Customer Care
form
You :
Bot
10:08 AM
5/30/2020

6. Result


Smart Internz
Application Details - IBM Cloud
Node-RED : node-red-mzolm.eu
Node-RED Dashboard
node-red-mzolm.eu-gb.mybluemix.net/ui/#!/0?socketid=RTS4Rpgk_DCITZWVAAAB
Node-RED Dashboard
Chatboat
hi
SUBMIT
CANCEL
You : hi
Bot
It's a beautiful morning. Hello
10:10 AM
5/30/2020


Smart Internz
Application Details - IBM Cloud
Node-RED : node-red-mzolm.eu
Node-RED Dashboard
node-red-mzolm.eu-gb.mybluemix.net/ui/#!/0?socketid=RTS4Rpgk_DCITZWVAAAB
Node-RED Dashboard
Chatboat
Enter your input *
hello
SUBMIT
CANCEL
You : hello
Bot
Hello. Good morning
10:11 AM
5/30/2020





Thus the Intelligent Customer Help Desk With Smart Document Understanding was successfully built and deployed. { https://node-red-mzolm.eu-gb.mybluemix.net/ui }

7. Advantages and Disadvantages

Advantages:

- Only a small amount of queries need representatives, hence the custom er is also satisfied with quick and easy solution.
- Companies can use these to decrease the work flow to the representatives.
- Reduce the number of reps.
- Cost Efficient.
- Decrease in the number of calls diverted to representatives.
- Less work load on employees.

Disadvantages:

- Sometimes the chatbot misleads the customers.
- The discovery returns wrong results when not properly configured.
- Giving same answer for different sentiments.
- Sometimes is unable to connect the customer sentiments and intents.

8. Application

- It can be deployed in popular social media applications like Facebook, Slack a nd Telegram because of the preexisting integrations with IBM Cloud Services.
- The chatbot can be deployed in any website to clear the basic doubts of the customer.
- It can be used as a Customer Helpdesk for small scale products as their manu al usually has the solution for the user's problems.

9. Conclusion

By following the above-mentioned steps, we create a basic chatbot which can help us to answer the basic questions of the customer or user related to location of the office, working hours and the information about the product. We have successfully created the intelligent help desk smart chatbot using IBM Watson Assistant, IBM Cloud Function, IBM Watson Discovery and Node-Red.

10. Future Scope

We can import the pre-built node-red flow and can improve our UI, moreover we can make a data base and use it to show the recent chats to the customer. We can also improve the results of discovery by enriching it with more fields and doing the Smart Data Annotation more accurately. We can get the premium version to increase the scope of our chatbot in terms of the calla and requests. We can also include Watson text to audio and Speech to text services to access the chatbot hands free. These are few of the future scopes which are possible.

11. Appendix

11.1. Source Code

11.1.1. IBM Cloud Functions Action (Node.js 10)

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

11.1.2 Node Red flow(.json) (formatted)


```
[
    {
        "id": "a50be1fb.5c75b",
        "type": "tab",
        "label": "Flow 1",
        "disabled": false,
        "info": ""
    },
    {
        "id": "69c2c85b.f47d48",
        "type": "ui_tab",
        "z": "",
        "name": "Customer Care",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    },
    {
        "id": "d346fcaa.6d87f",
        "type": "ui_group",
        "z": "",
        "name": "Chatboat",
        "tab": "69c2c85b.f47d48",
        "order": 1,
        "disp": true,
        "width": 15,
        "collapse": false
    },
    {
        "id": "a6ace9b5.11d4b8",
```

```
  "type": "ui_base",
  "theme": {
    "name": "theme-light",
    "lightTheme": {
      "default": "#0094CE",
      "baseColor": "#8000ff",
      "baseFont": "Times New Roman,Times,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",
      "reset": false
    },
    "themeState": {
      "base-color": {
        "default": "#0094CE",
        "value": "#8000ff",
        "edited": true
```

```
},
"page-titlebar-backgroundColor": {
   "value": "#8000ff",
   "edited": false
},
"page-backgroundColor": {
   "value": "#fafafa",
   "edited": false
},
"page-sidebar-backgroundColor": {
   "value": "#000000",
   "edited": false
},
"group-textColor": {
   "value": "#a64dff",
   "edited": false
},
"group-borderColor": {
   "value": "#ffffff",
   "edited": false
},
"group-backgroundColor": {
   "value": "#ffffff",
   "edited": false
},
"widget-textColor": {
   "value": "#111111",
   "edited": false
},
"widget-backgroundColor": {
   "value": "#8000ff",
```

```
            "edited": false
        },
        "widget-borderColor": {
            "value": "#ffffff",
            "edited": false
        },
        "base-font": {
            "value": "Times New Roman,Times,serif"
        }
    },
    "angularTheme": {
        "primary": "indigo",
        "accents": "blue",
        "warn": "red",
        "background": "grey"
    }
},
"site": {
    "name": "Node-RED Dashboard",
    "hideToolbar": "false",
    "allowSwipe": "false",
    "lockMenu": "true",
    "allowTempTheme": "true",
    "dateFormat": "DD/MM/YYYY",
    "sizes": {
        "sx": 30,
        "sy": 37,
        "gx": 0,
        "gy": 0,
        "cx": 16,
        "cy": 16,
```

```
            "px": 0,
            "py": 0
          }
        }
    },
    {
        "id": "3530ddaf.f6d482",
        "type": "ui_tab",
        "name": "Tab 2",
        "icon": "dashboard",
        "order": 2
    },
    {
        "id": "6992a746.8ff768",
        "type": "watson-conversation-v1",
        "z": "a50be1fb.5c75b",
        "name": "Customer Care",
        "workspaceid": "1edd8c1f-18e1-43e8-8dc3-709bc13a8914",
        "multiuser": false,
        "context": true,
        "empty-payload": false,
        "service-endpoint": "https://api.eu-
gb.assistant.watson.cloud.ibm.com/instances/0a003d44-1846-432a-ac0d-
fd8d2ea76a3e",
        "timeout": "",
        "optout-learning": false,
        "x": 560,
        "y": 160,
        "wires": [
          [
            "403d959b.941acc",
```

```
            "a36765c3.e52628"
        ]
    ]
},
{
    "id": "36945a85.7422f6",
    "type": "ui_text",
    "z": "a50be1fb.5c75b",
    "group": "d346fcaa.6d87f",
    "order": 3,
    "width": 15,
    "height": 8,
    "name": "",
    "label": "Bot",
    "format": "{{msg.payload}}",
    "layout": "col-center",
    "x": 800,
    "y": 440,
    "wires": []
},
{
    "id": "727b545b.8890cc",
    "type": "ui_text",
    "z": "a50be1fb.5c75b",
    "group": "d346fcaa.6d87f",
    "order": 2,
    "width": 0,
    "height": 0,
    "name": "",
    "label": "You :",
    "format": "{{msg.payload}}",
```

```
    "layout": "row-left",
    "x": 330,
    "y": 460,
    "wires": []
},
{
    "id": "a3dac9fe.b52f38",
    "type": "function",
    "z": "a50be1fb.5c75b",
    "name": "",
    "func": "msg.payload=msg.payload.text;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 330,
    "y": 280,
    "wires": [
      [
        "6992a746.8ff768",
        "727b545b.8890cc"
      ]
    ]
},
{
    "id": "403d959b.941acc",
    "type": "function",
    "z": "a50be1fb.5c75b",
    "name": "",
    "func": "msg.payload=msg.payload.output.text[0];\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 770,
```

```
    "y": 280,
    "wires": [
      [
        "36945a85.7422f6"
      ]
    ]
  },
  {
    "id": "a36765c3.e52628",
    "type": "debug",
    "z": "a50be1fb.5c75b",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "false",
    "x": 900,
    "y": 100,
    "wires": []
  },
  {
    "id": "29abb804.e31a48",
    "type": "ui_form",
    "z": "a50be1fb.5c75b",
    "name": "",
    "label": "",
    "group": "d346fcaa.6d87f",
    "order": 1,
    "width": 0,
    "height": 0,
```

```
    "options": [
      {
        "label": "Enter your input",
        "value": "text",
        "type": "text",
        "required": true,
        "rows": null
      }
    ],
    "formValue": {
      "text": ""
    },
    "payload": "",
    "submit": "submit",
    "cancel": "cancel",
    "topic": "",
    "x": 110,
    "y": 380,
    "wires": [
      [
        "a3dac9fe.b52f38"
      ]
    ]
  }
]
```

11.2 References

- https://www.ibm.com/cloud/architecture/tutorials/cognitive_discovery
- https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started
- https://developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chatbot-on-nodered/
- http://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red
- https://github.com/IBM/watson-discovery-sdu-with-assistant