Intelligent Customer Help Desk with Smart Document Understanding

Name: RUDRA PRATAP SINGH

Category: Machine Learning

Mail id: rr4737@srmist.edu.in

College: SRM Institute Of Science and Technology

Internship At Smartinternz.com@2020

INDEX

1. INTRODUCTION

1. Overview:

Here we are going to prepare a chat bot by the use of ibm cloud services and the Watson services like assistant and discovery.

Services we are using in making chat bot are Discovery , Assistant, Cloud function and Node Red. By the end of the project, we'll learn how to make chat bot and best practices of integration of Watson services, and how they can build interactive information retrieval systems with Discovery and Assistant with the combination of functions and integrated to node red.

Project Requirements: Python, IBM Cloud, IBMWatson

Functional Requirements: IBMcloud

Technical Requirements: AI,ML,WATSONAI,PYTHON

Software Requirements: Watson assistant, Watsondiscovery,node-red.

Project Deliverables: Smartinternz Intership

Project Team: Garimella Sri Krishna Priya

Project Duration:19days

2. PURPOSE :

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

In this project, there will be another option. If the customer question is about the operation of a device, the application shall pass the question onto Watson Discovery Service, which has been

pre-loaded with the device's owners manual. So now, instead of "Would you like to speak to a customer representative?" we can return relevant sections of the owners manual to help solve our customers'problems.

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.

i. Scope of Work
1. Create a customer care dialog skill in WatsonAssistant.

2. Use Smart Document Understanding to build an enhanced Watson Discovery collection.

3. Create an IBM Cloud Functions web action that allows Watson Assistant to post queries to WatsonDiscovery.

4. Build a web application with integration to all these services & deploy the same on IBM CloudPlatform.

2. LITERATURE SURVEY

1. Existing problem:

A good customer Chat bot should minimize involvement of customer agent to chat with customer to clarify his/her doubts. So to achieve this we should include an virtual agent in chat bot so that it will take care of real involvement of customer agent and customer can clarifies his doubts with fast chat bots. In general we can say that the chat bot is the platform where the user can clarify his various doubts about the product which he/she want to purchase and they have the full rights to know about the product. Here we have to clarify the user doubts without the help of the agent and if the chat bot can't answer the queries the they can contact to the agent or any other person.

2. Proposed solution:

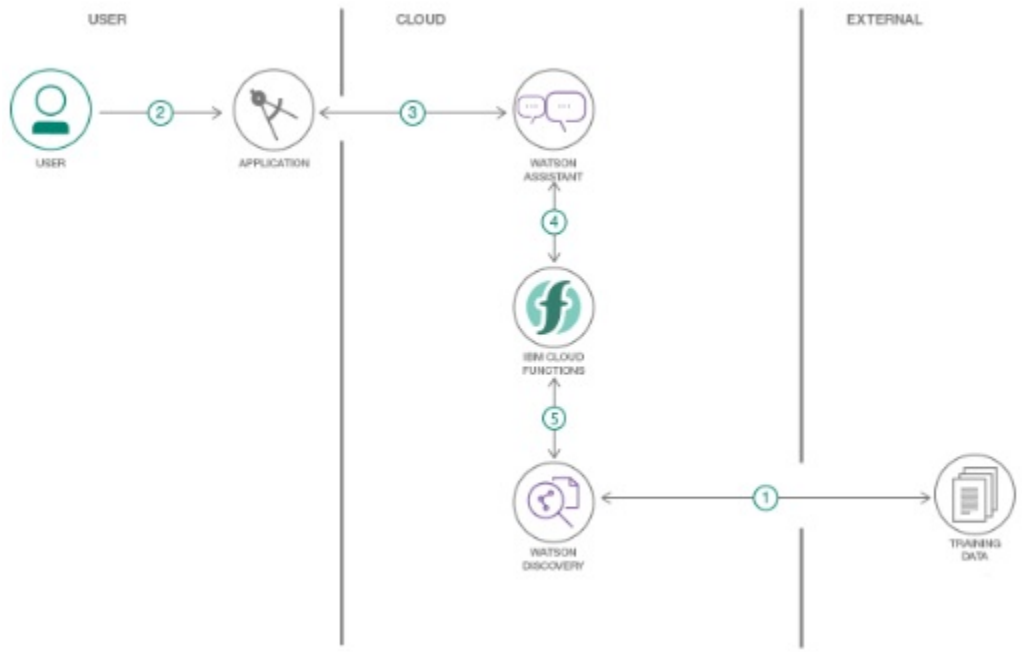
For the above problem to get solved we have to put an virtual agent in chat bot so it can understand

the queries that are posted by customers. The virtual agent should trained from some insight records based company background so it can answer queries based on the product or related to company. In this project I have used Watson Discovery to achieve the above solution. And later including Assistant and Discovery on Node-RED.

We use function to integrate the discovery and then the assistant and then the integration of all these are done in node-red. The discovery contains the manuals where we train them and then with the help of functions we integrate it to assistant.

3. THEORITICAL ANALYSIS

1-Block/FlowDiagram:



1. The document is annotated using Watson Discovery SDU.

2. The user interacts with the backend server via the app UI. The front end app UI is a chat bot 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 product operation question, 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.

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

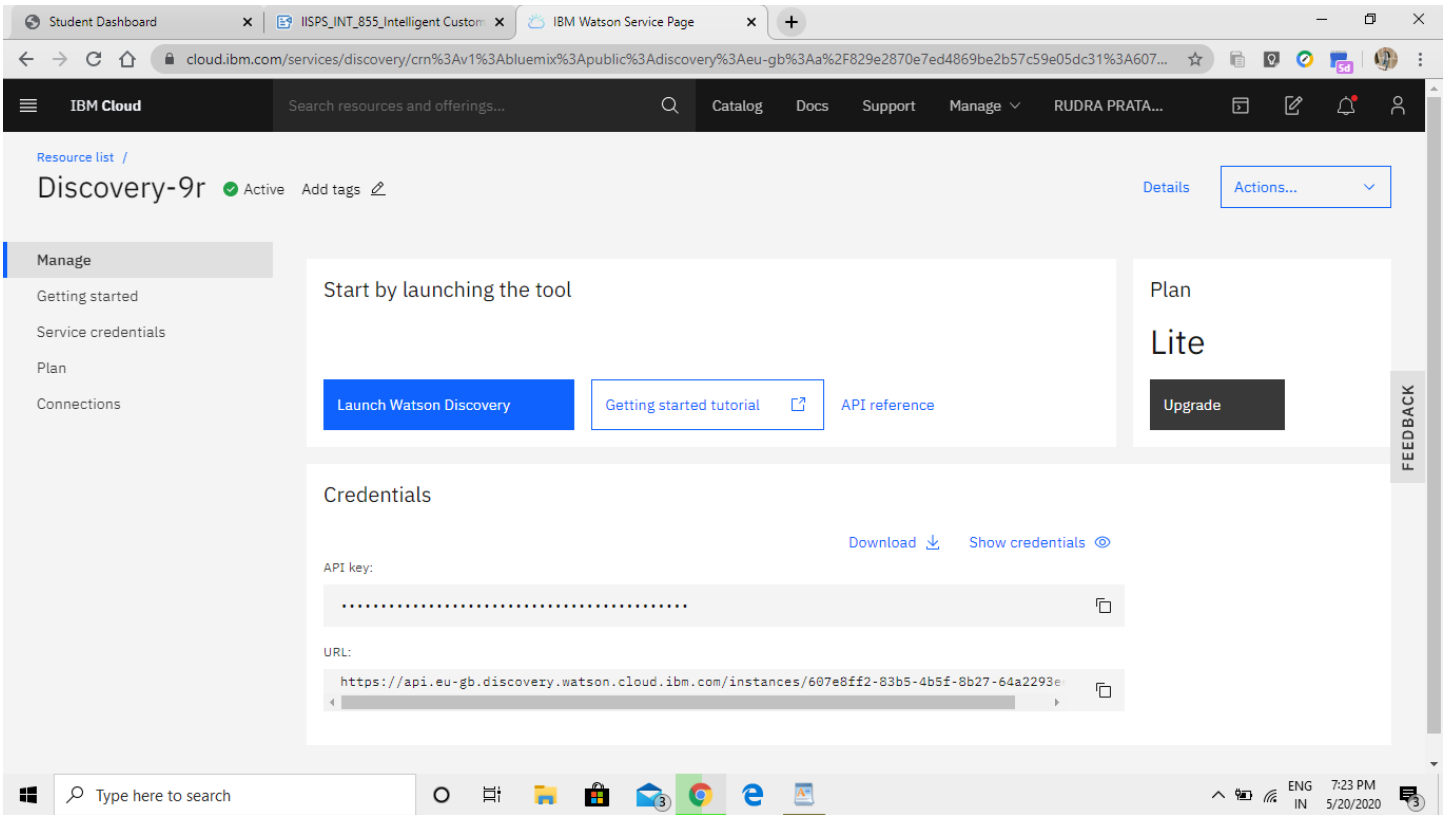
6. Deploy and run Node Red app.

4. EXPERIMENTAL INVESTIGATIONS:

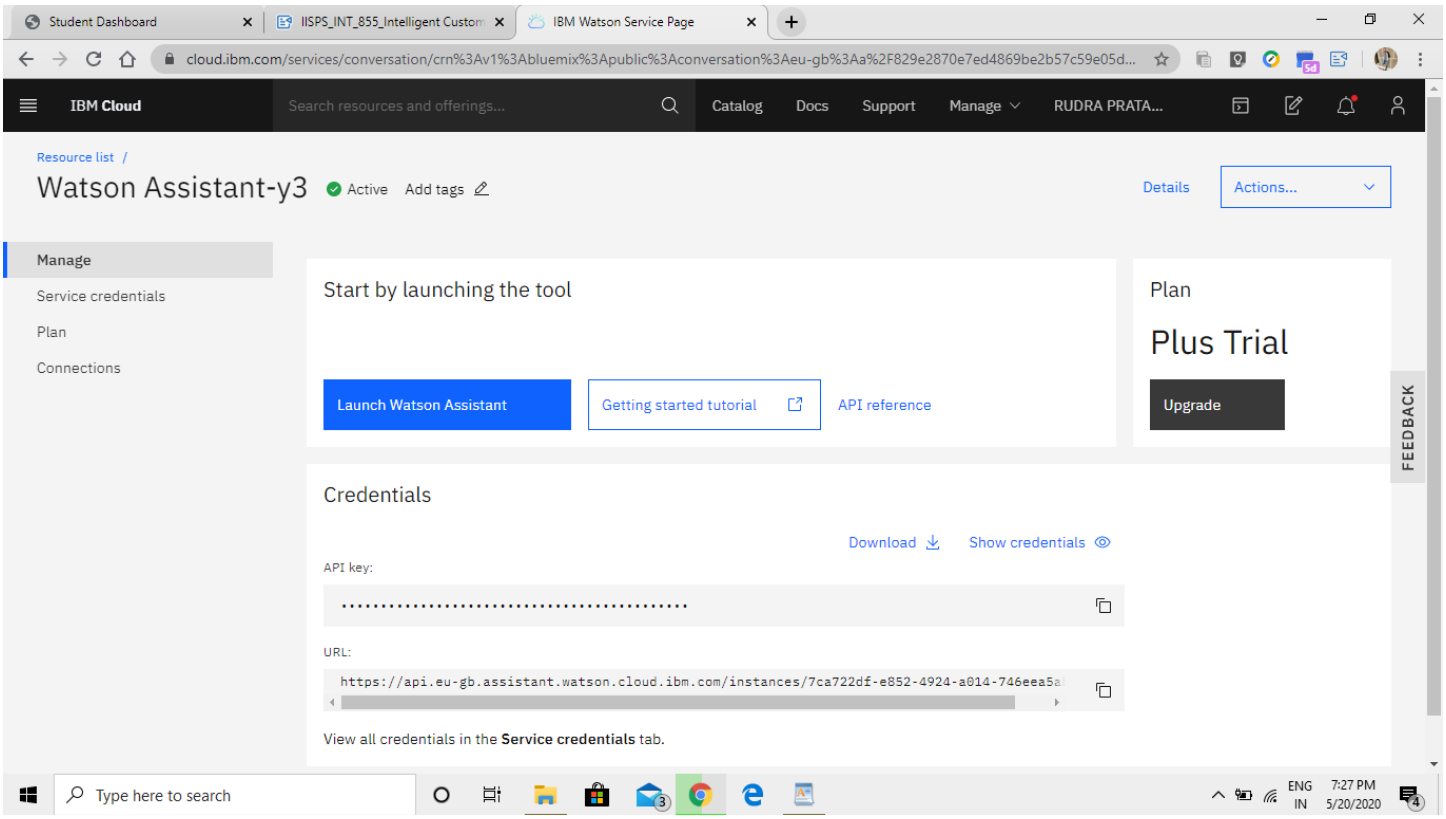
1.Create IBM Cloud services

Create the following services:

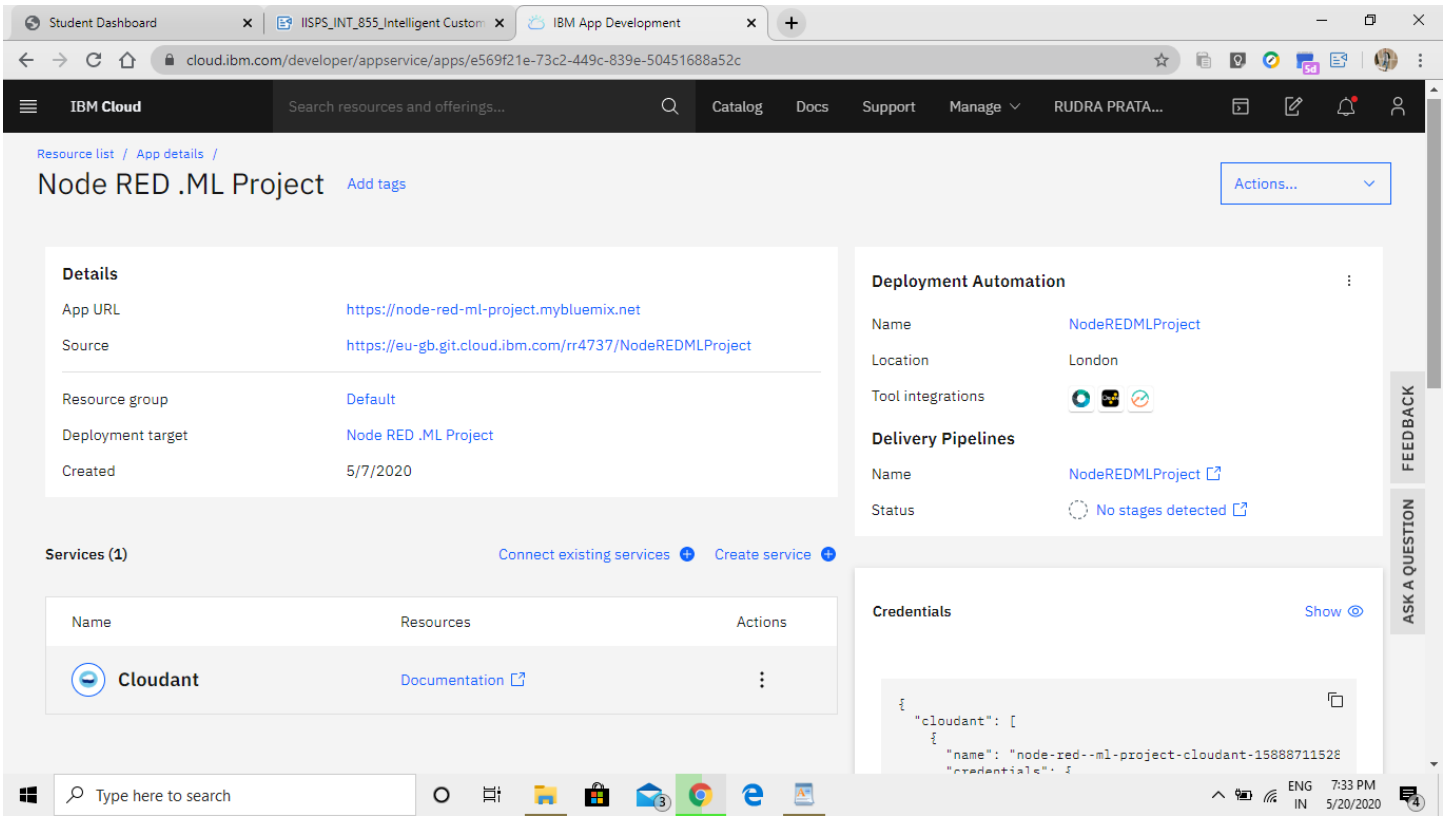
a. *Watson Discovery* :



b. WatsonAssistant :



c. Node-Red:



2. Configure WatsonDiscovery

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 ecobee3_UserGuide.pdf file located in the data directory of your local repo.

The Ecobee is a popular residential thermostat that has a wifi interface and multiple configuration options.

Before applying SDU to our document, lets do some simple queries on the data so that we can

compare it to results found after applying SDU.

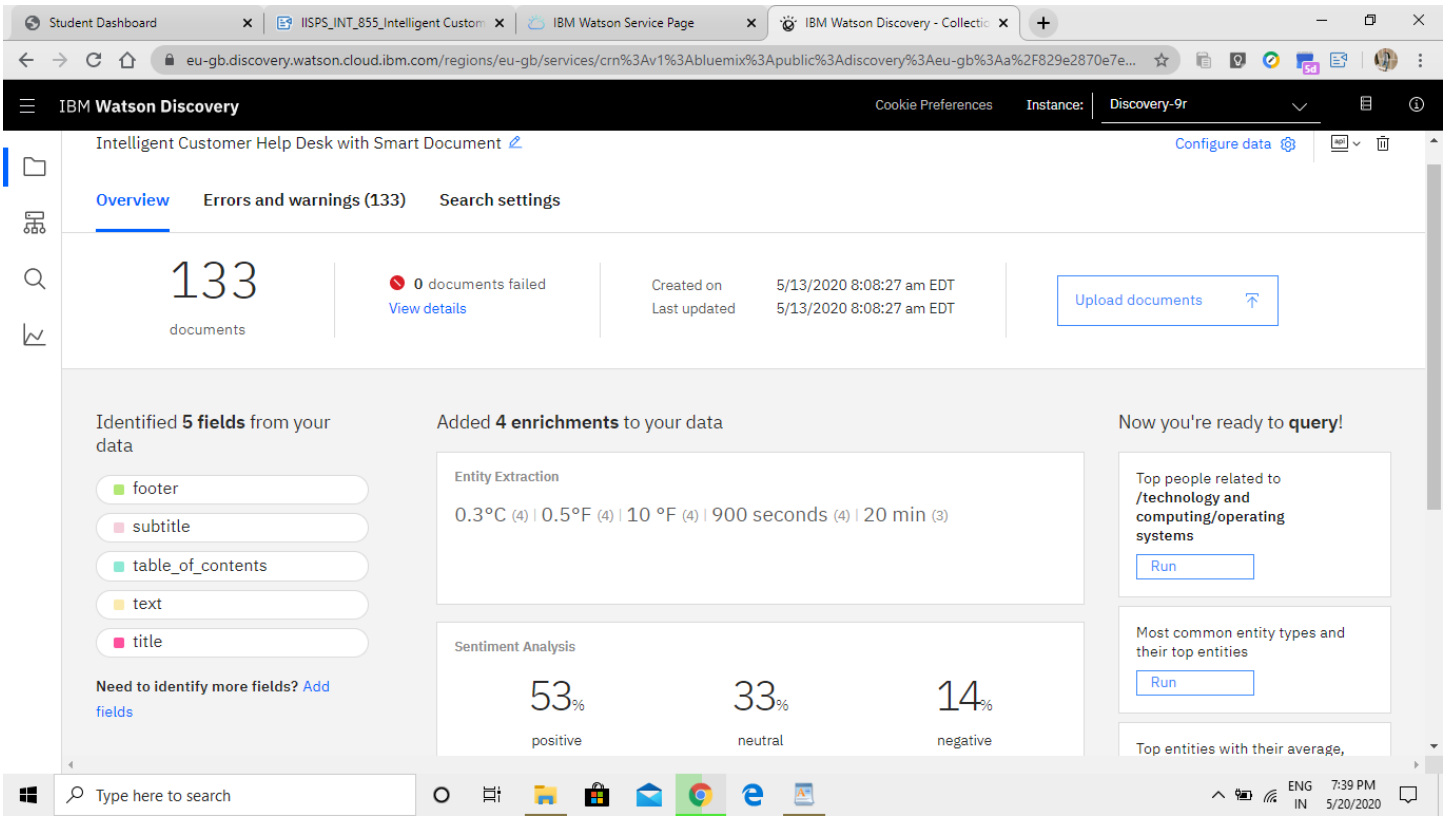
Click the Build your own query [1] button.

Enter queries related to the operation of the thermostat and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question.

Annotate with SDU

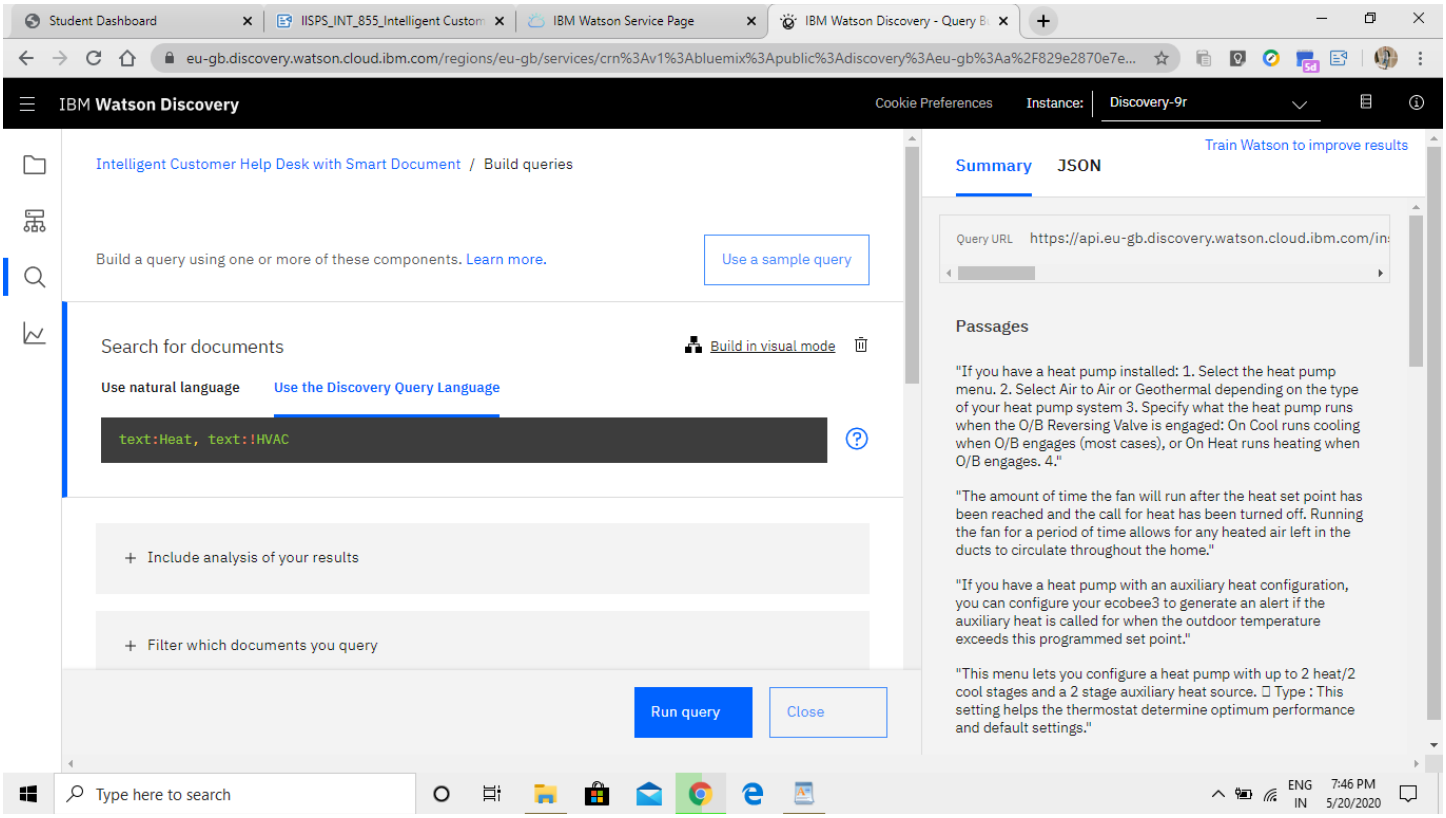
Now let's apply SDU to our document to see if we can generate some better query responses.From the Discovery collection panel, click the Configure data button (located in the top right corner) to start the SDUprocess.

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



Click the Build your own query [1] button.

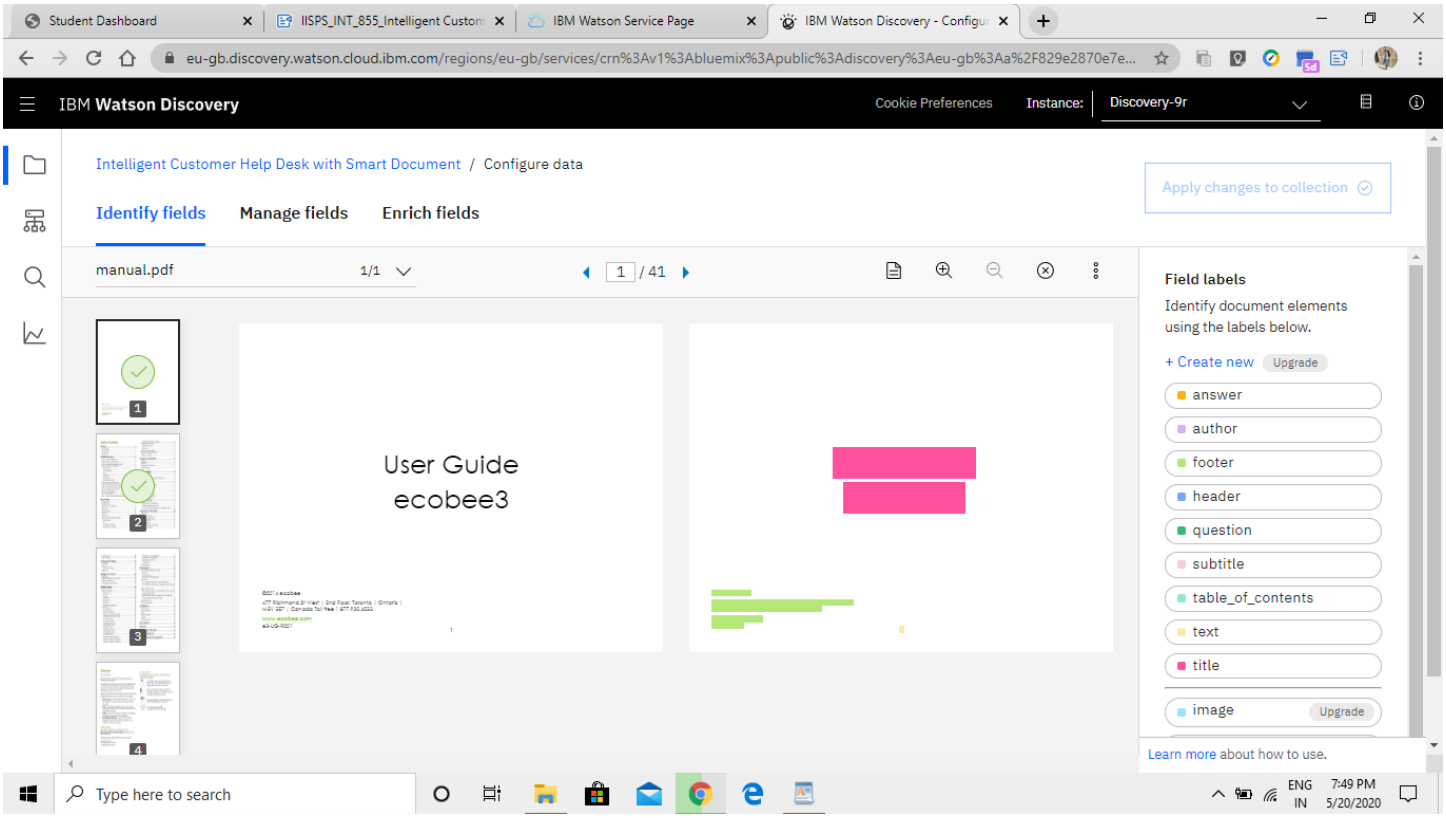
Enter queries related to the operation of the thermostat and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question.



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 SDUprocess.

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.

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

is the current page being annotated.

is where you select text and assign it a label.

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

Click [5] to submit the page to Discovery.

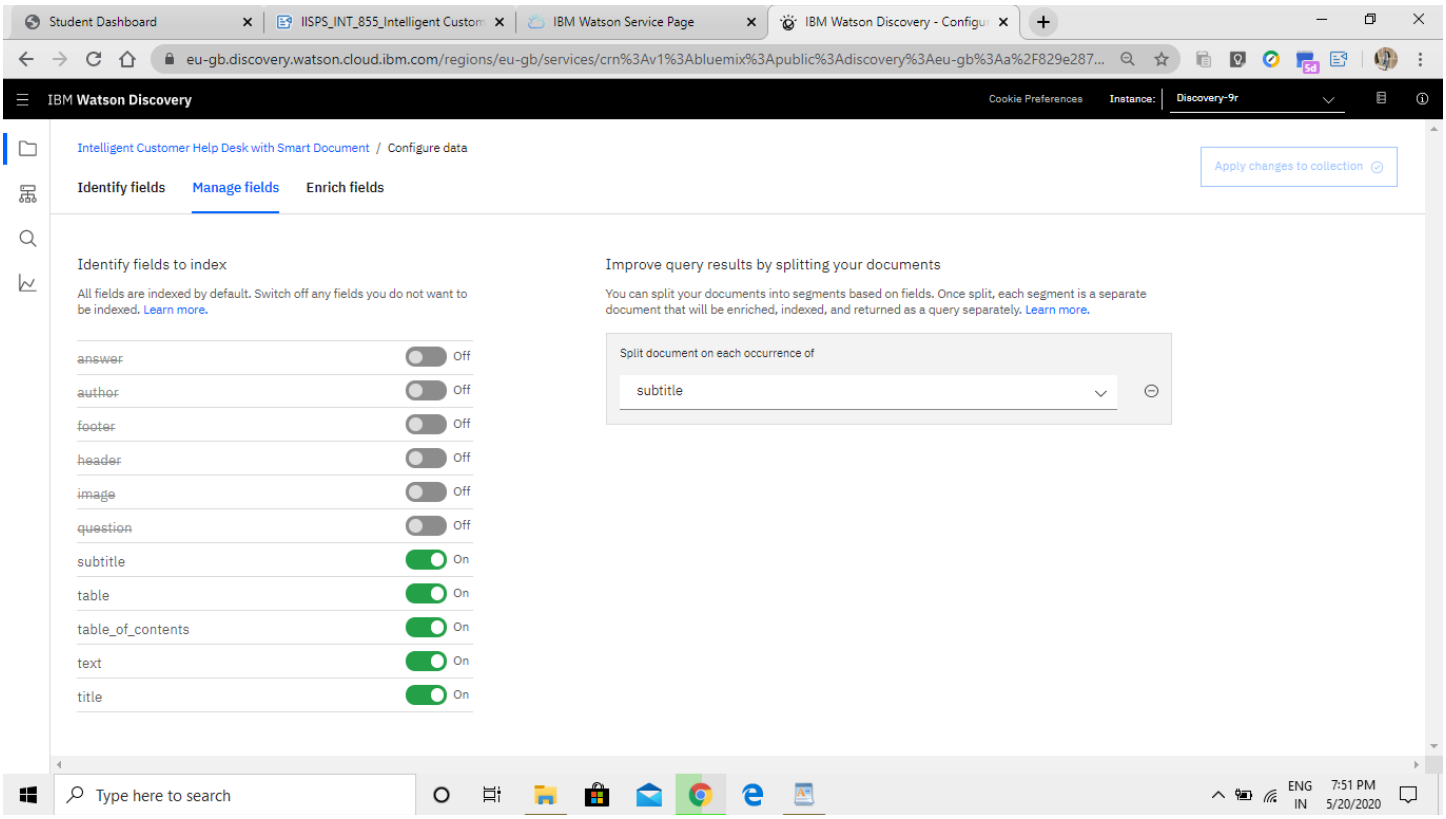
Click [6] when you have completed the annotation process.

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
The table of contents (shown in the first few pages) as table_of_contents All headers and sub-headers (typed in light green text) as a 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.



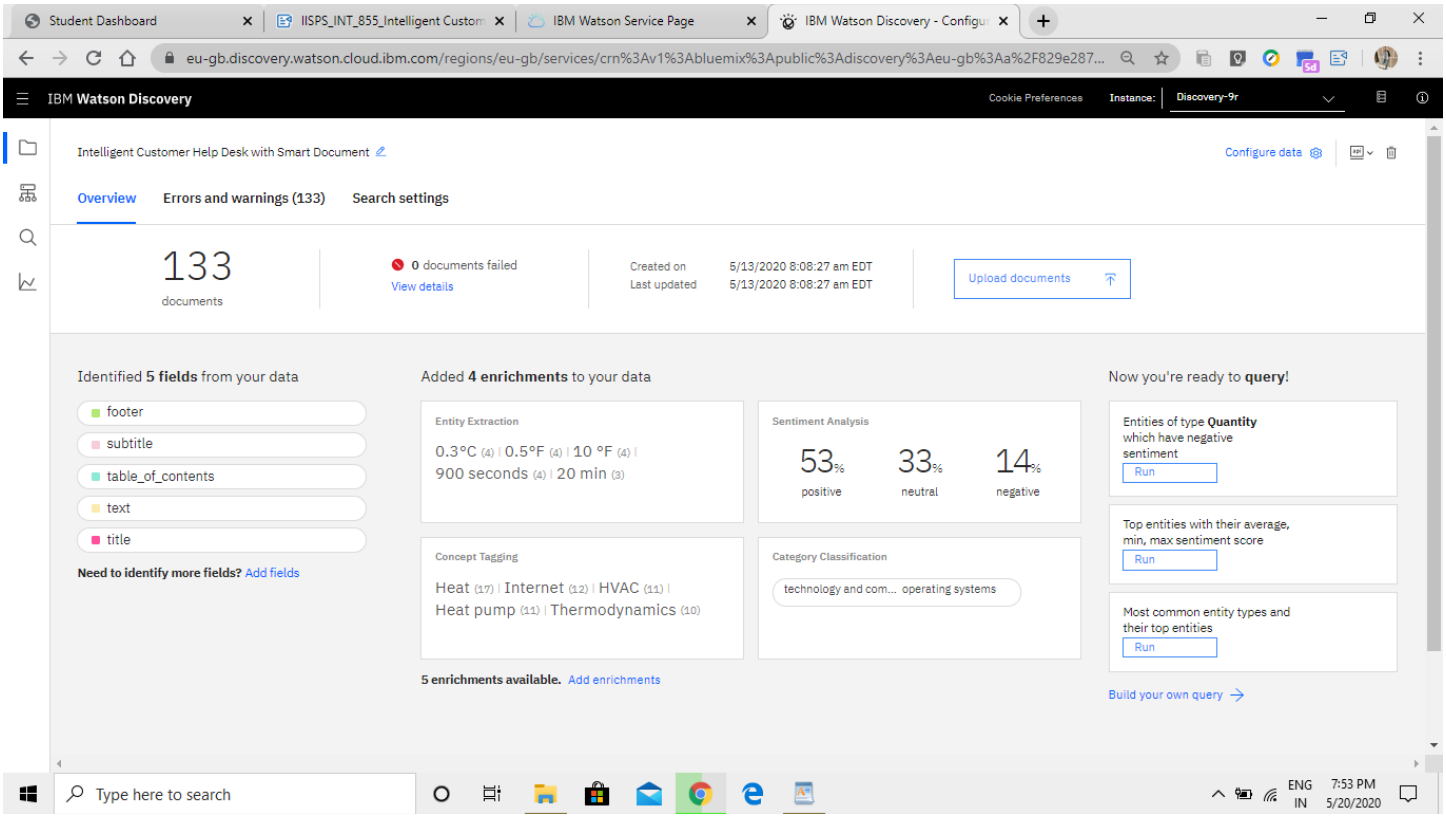
Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except subtitles and text.

is telling Discovery to split the document apart, based on subtitle.

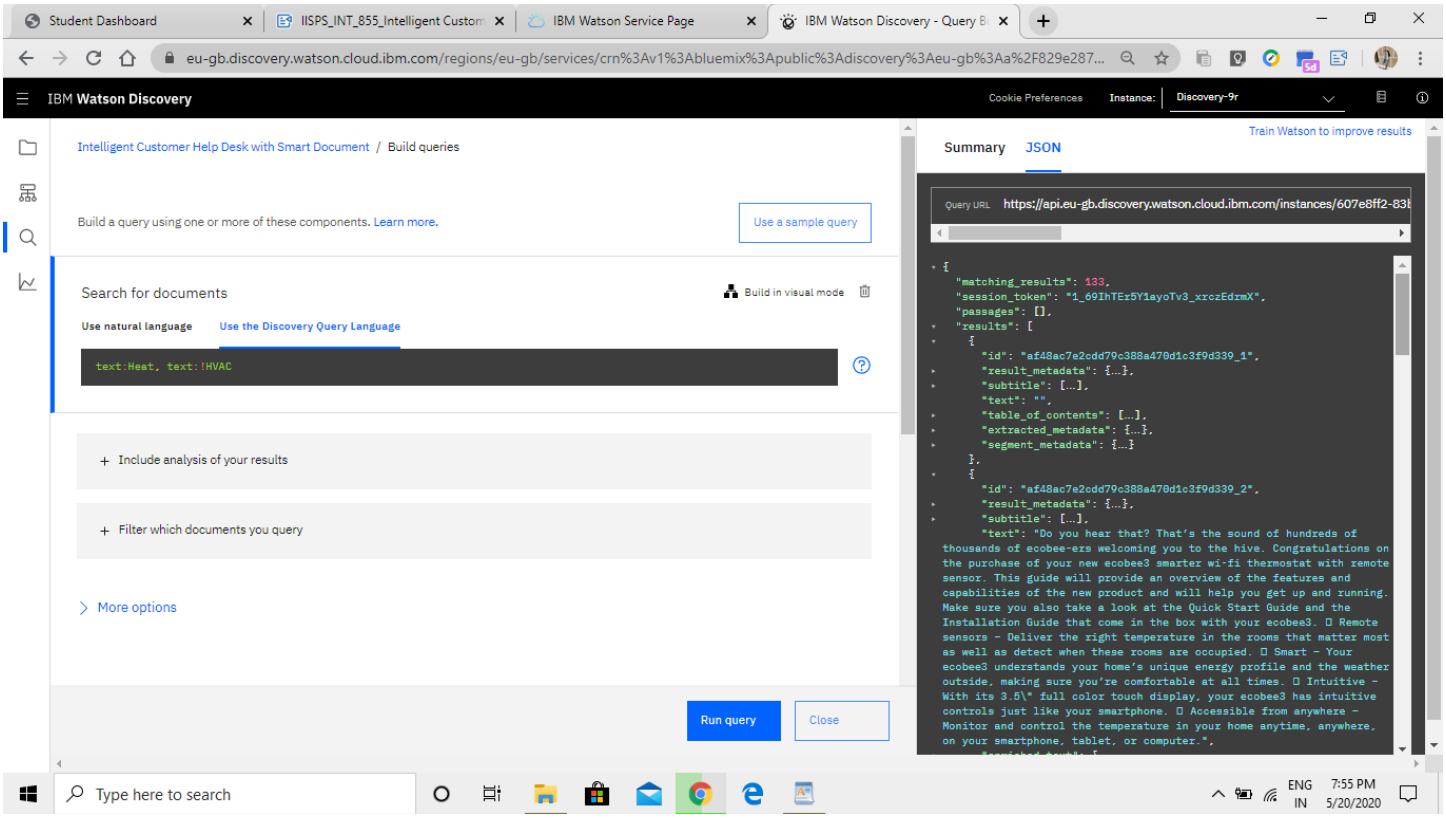
Click [4] to submit your changes.

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

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



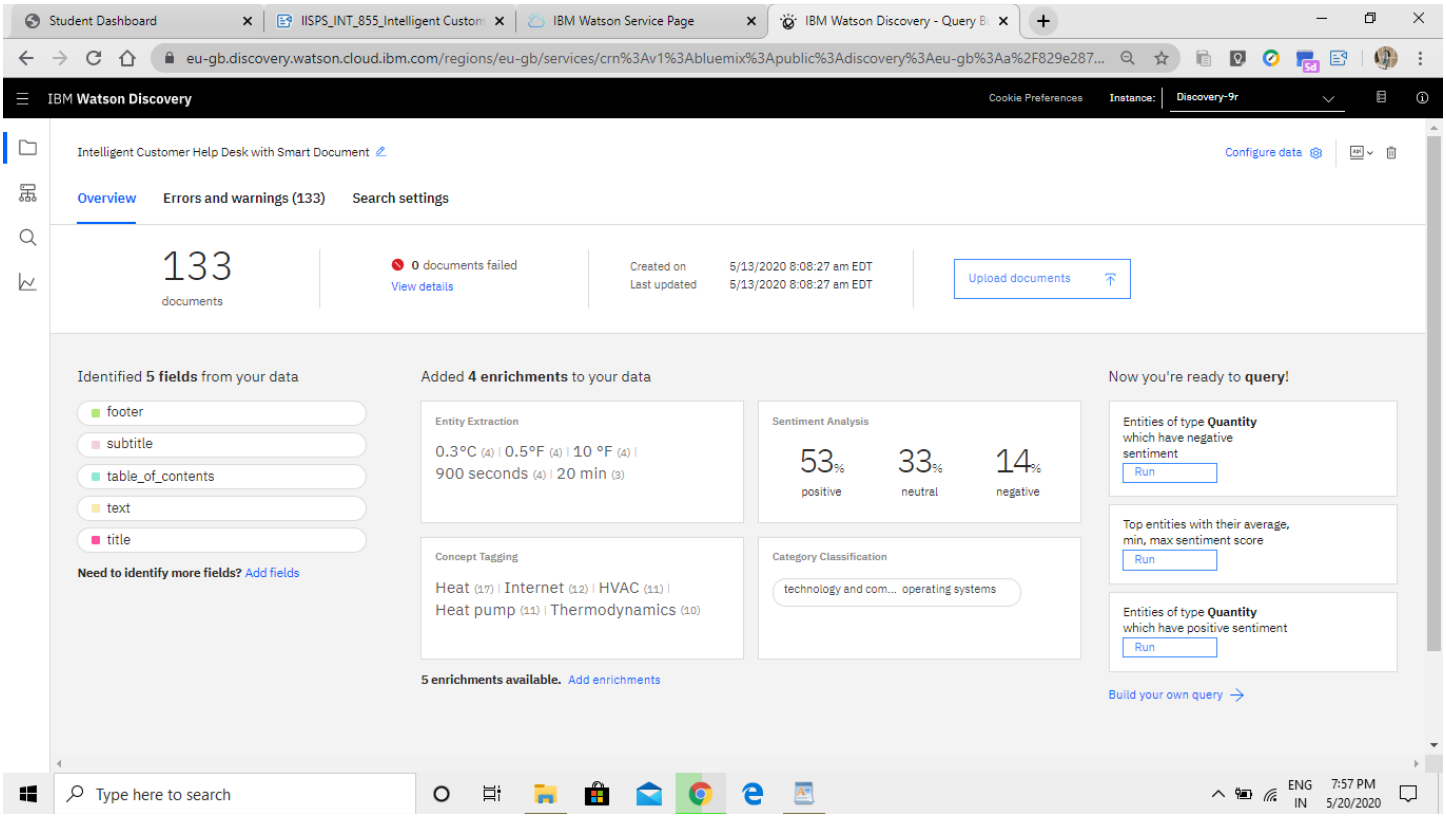
Return to the query panel (click Build your own query) and see how much better the results are.



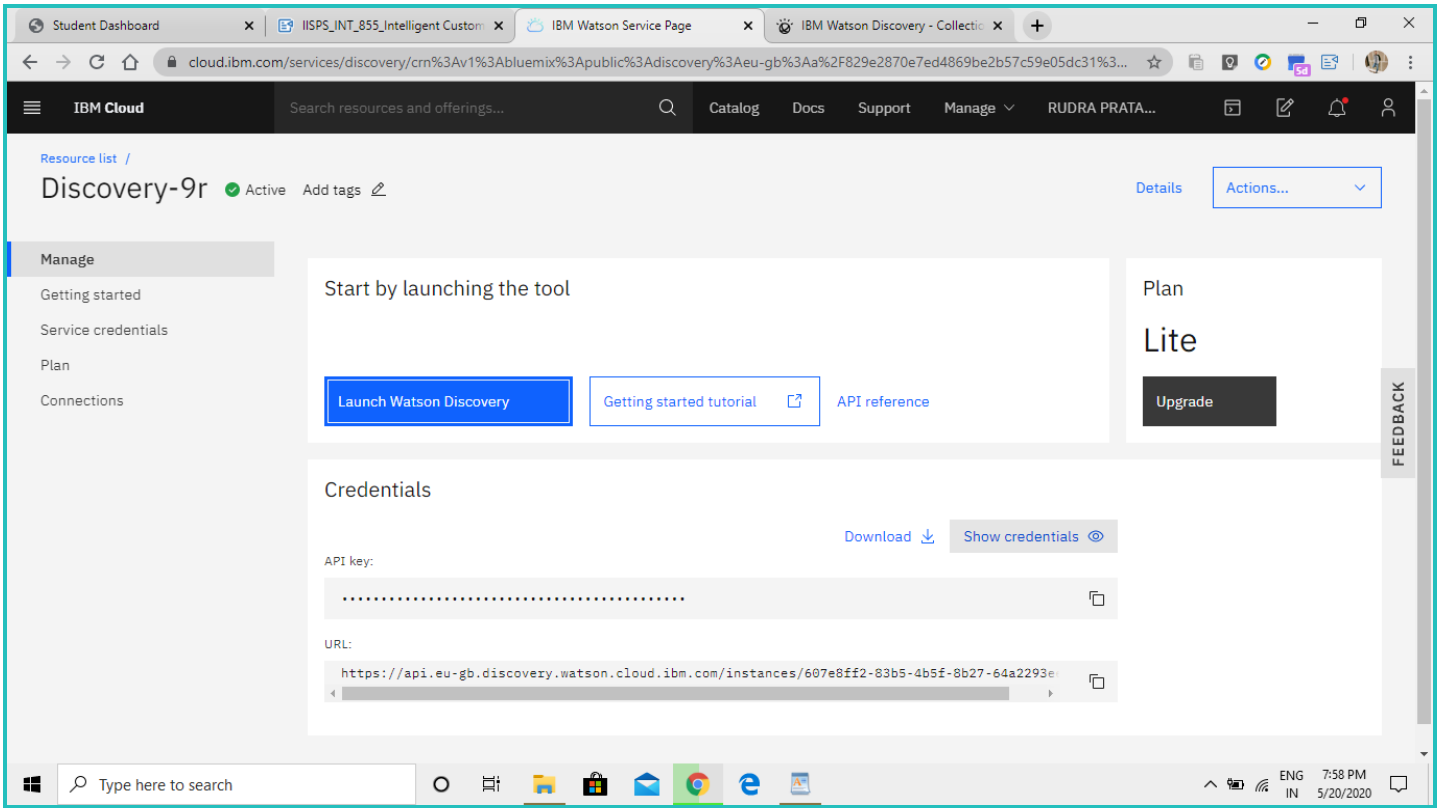
Store credentials for future use

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 and Environment ID values can be found by clicking the dropdown button [1] located at the top right side of your collection panel:



For credentials, return to the main panel of your Discovery service, and click the Service credentials tab.

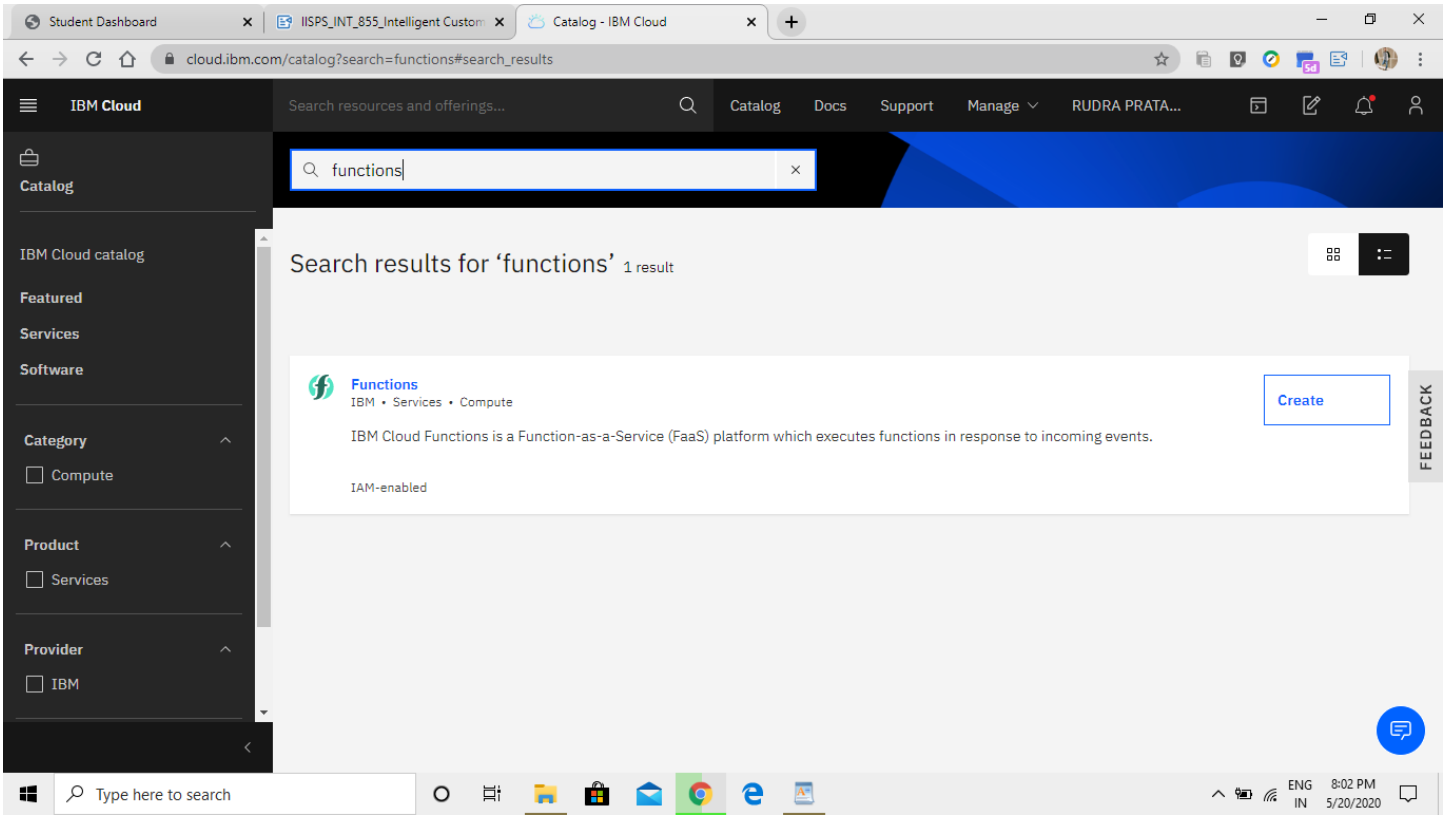


Click the View credentials [2] drop-down menu to view the IAM apikey [3] and URL endpoint [4] for your service.

3. Create IBM Cloud Functions action

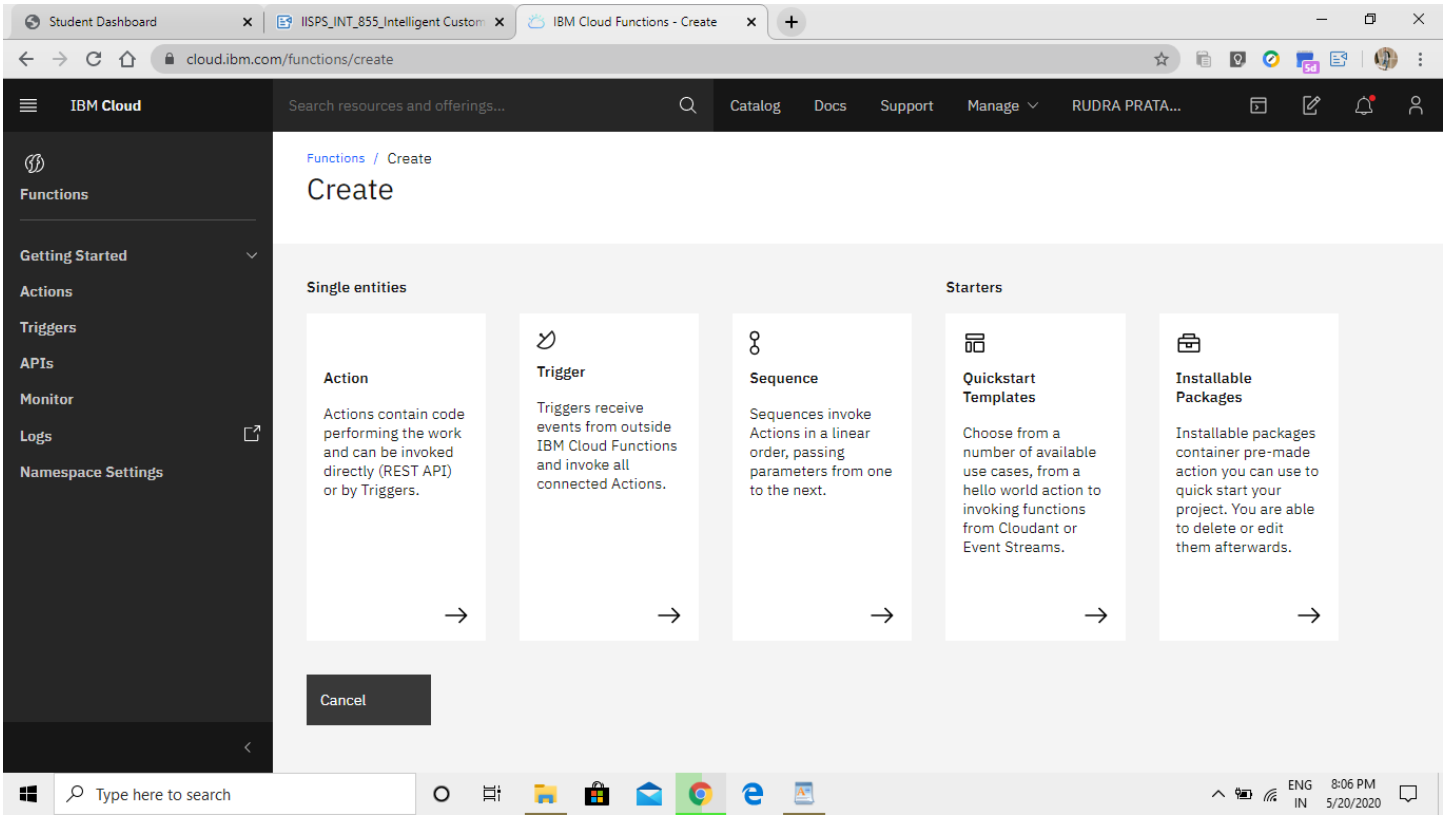
Now let's create the web action that will make queries against our Discovery collection.

Start the IBM Cloud Functions service by selecting Create Resource from the IBM Cloud dashboard. Enter functions as the filter [1], then select the Functions card [2]:

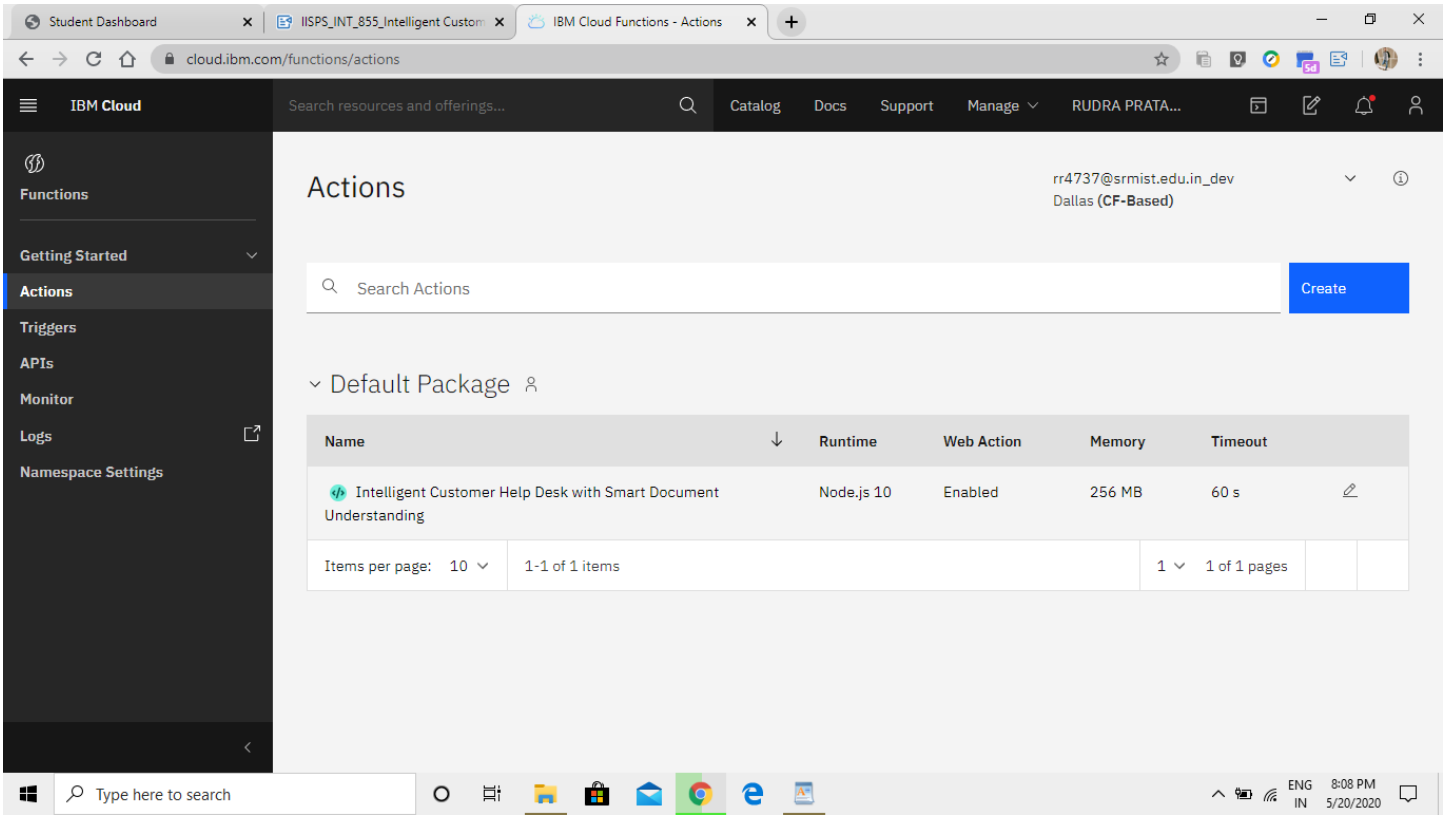


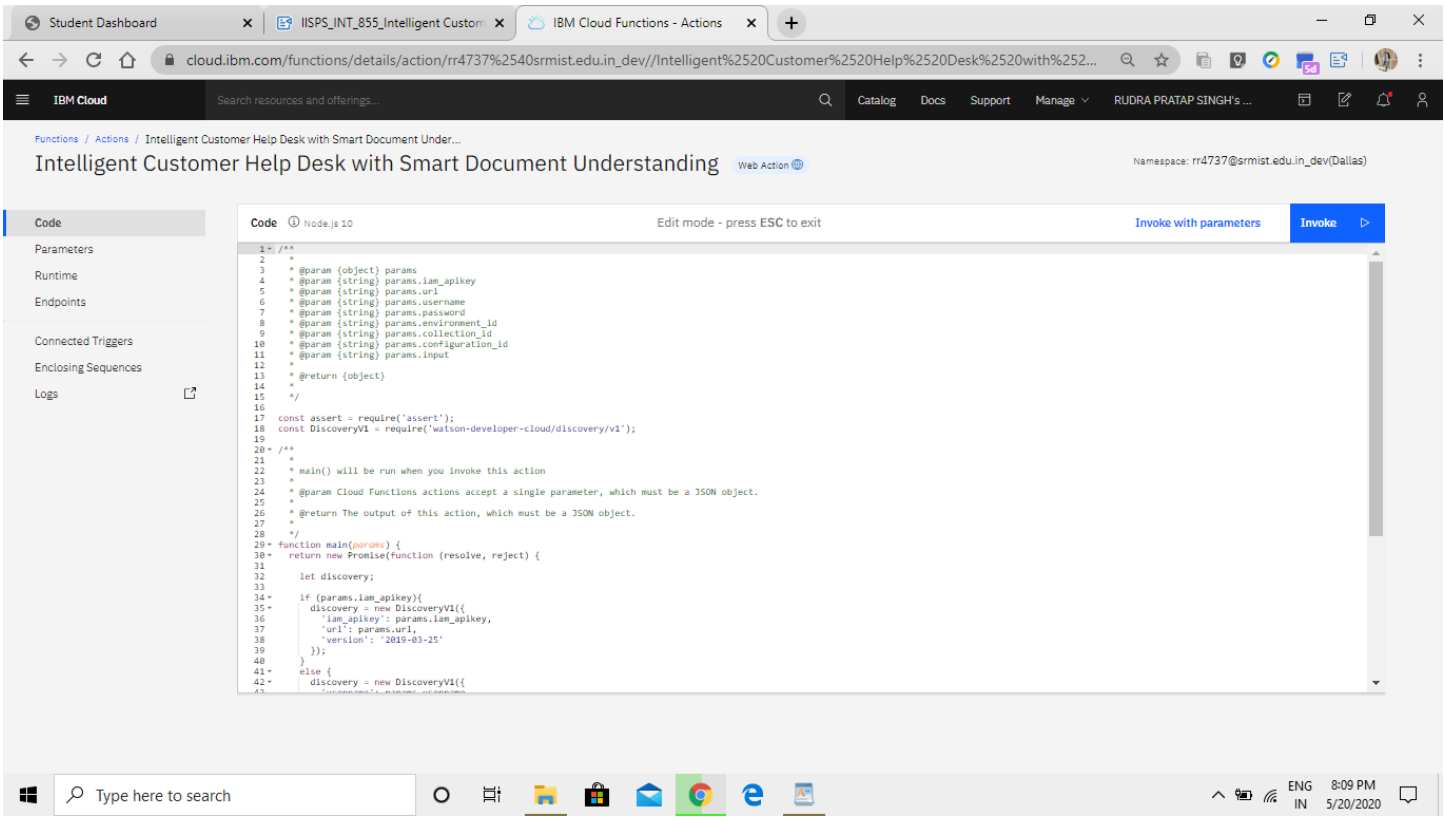
From the Functions main panel, click on the Actions tab. Then click on Create. From the Create panel, select the Create Action option.

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]:

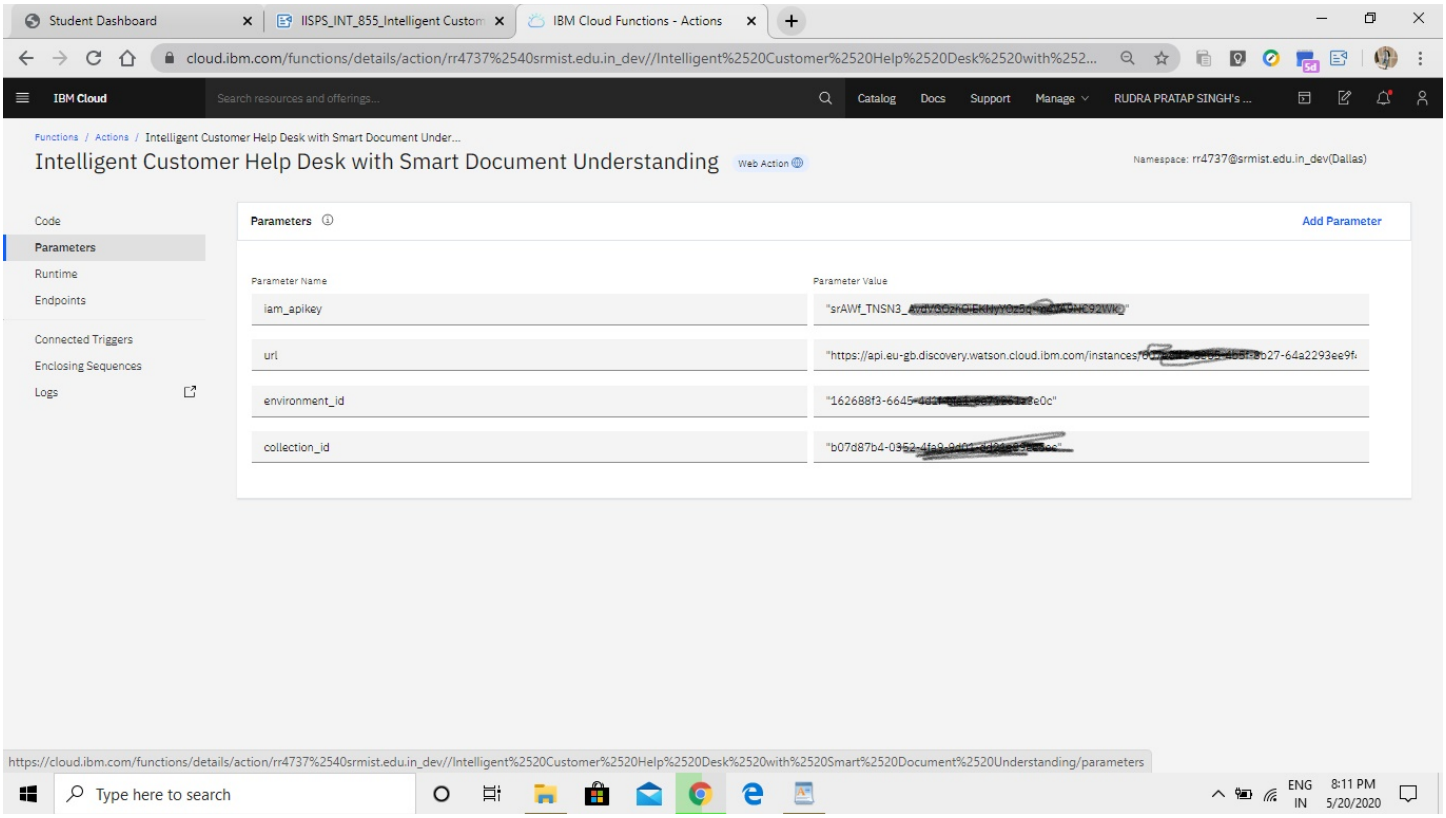

Student Dashboard
IISPS_INT_855_Intelligent Custom
IBM Cloud Functions - Actions
cloud.ibm.com/functions/actions
IBM Cloud
Search resources and offerings...
Catalog
Docs
Support
Manage
RUDRA PRATA...
Functions
Getting Started
Actions
Triggers
APIs
Monitor
Logs
Namespace Settings
Actions
rr4737@srmist.edu.in_dev
Dallas (CF-Based)
Search Actions
Create
Default Package
Name
Runtime
Web Action
Memory
Timeout
Intelligent Customer Help Desk with Smart Document Understanding
Node.js 10
Enabled
256 MB
60 s
Items per page: 10
1-1 of 1 items
1 of 1 pages
Type here to search
ENG
IN
8:08 PM
5/20/2020


Student Dashboard
IISPS_INT_855_Intelligent Custom
IBM Cloud Functions - Actions
cloud.ibm.com/functions/details/action/rr4737%2540srmist.edu.in_dev//Intelligent%2520Customer%2520Help%2520Desk%2520with%252...
IBM Cloud
Search resources and offerings...
Catalog
Docs
Support
Manage
RUDRA PRATAP SINGH's ...
Functions / Actions / Intelligent Customer Help Desk with Smart Document Under...
Intelligent Customer Help Desk with Smart Document Understanding
Web Action
Namespace: rr4737@srmist.edu.in_dev(Dallas)
Code
Parameters
Runtime
Endpoints
Connected Triggers
Enclosing Sequences
Logs
Code
Node.js 10
Edit mode - press ESC to exit
Invoke with parameters
Invoke
/**
* @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({
Type here to search
ENG
IN
8:09 PM
5/20/2020

In the code editor window [2], cut and paste in the code from the disco-action.js file found in the actions directory of your local repo. 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 [3], it will fail due to credentials not being defined yet. We'll do this next.

Select the Parameters tab [1]:

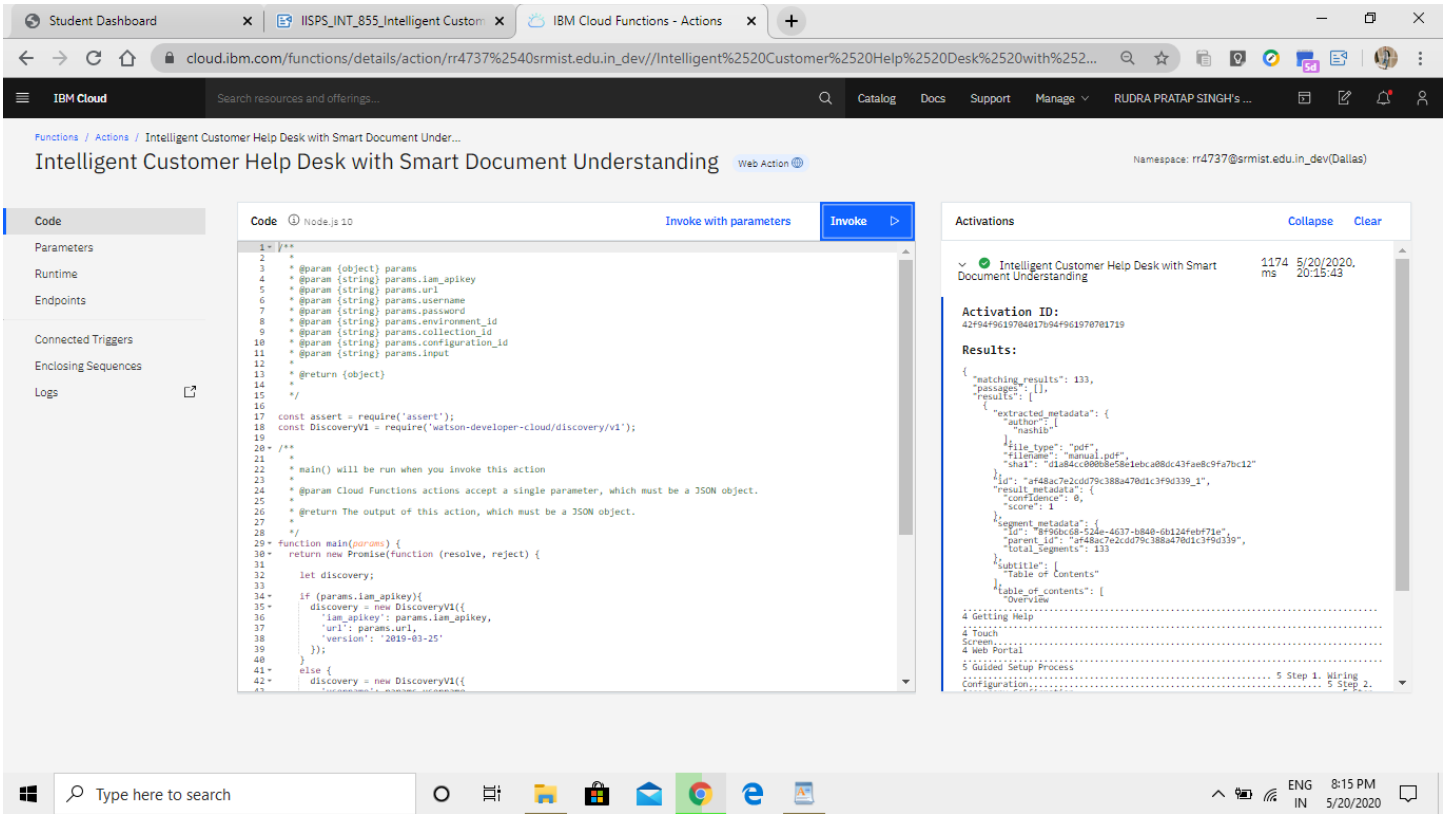


Add the following keys:

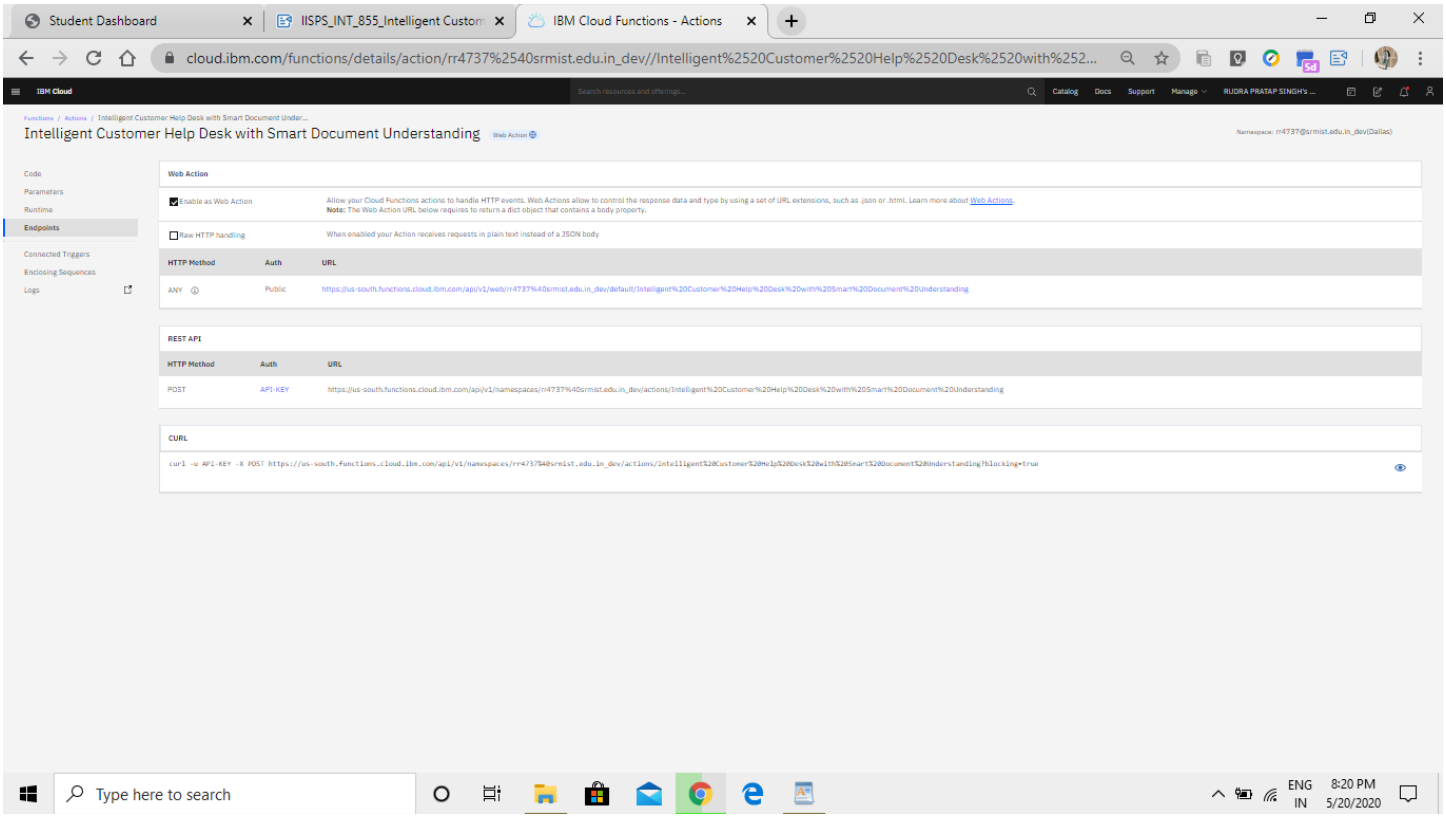
url
environment_id
collection_id
iam_apikey

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 Discoveryservice:



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.

To verify you have entered the correct Discovery parameters, execute the provided curl command [4]. If it fails, re-check your parameter values.

4. Configure Watson Assistant

Launch the Watson Assistant tool and create a new dialog skill. Select the Use sample skill option as your starting point.This dialog skill contains all of the nodes needed to have a typical call center conversation with a user.

Add new intent

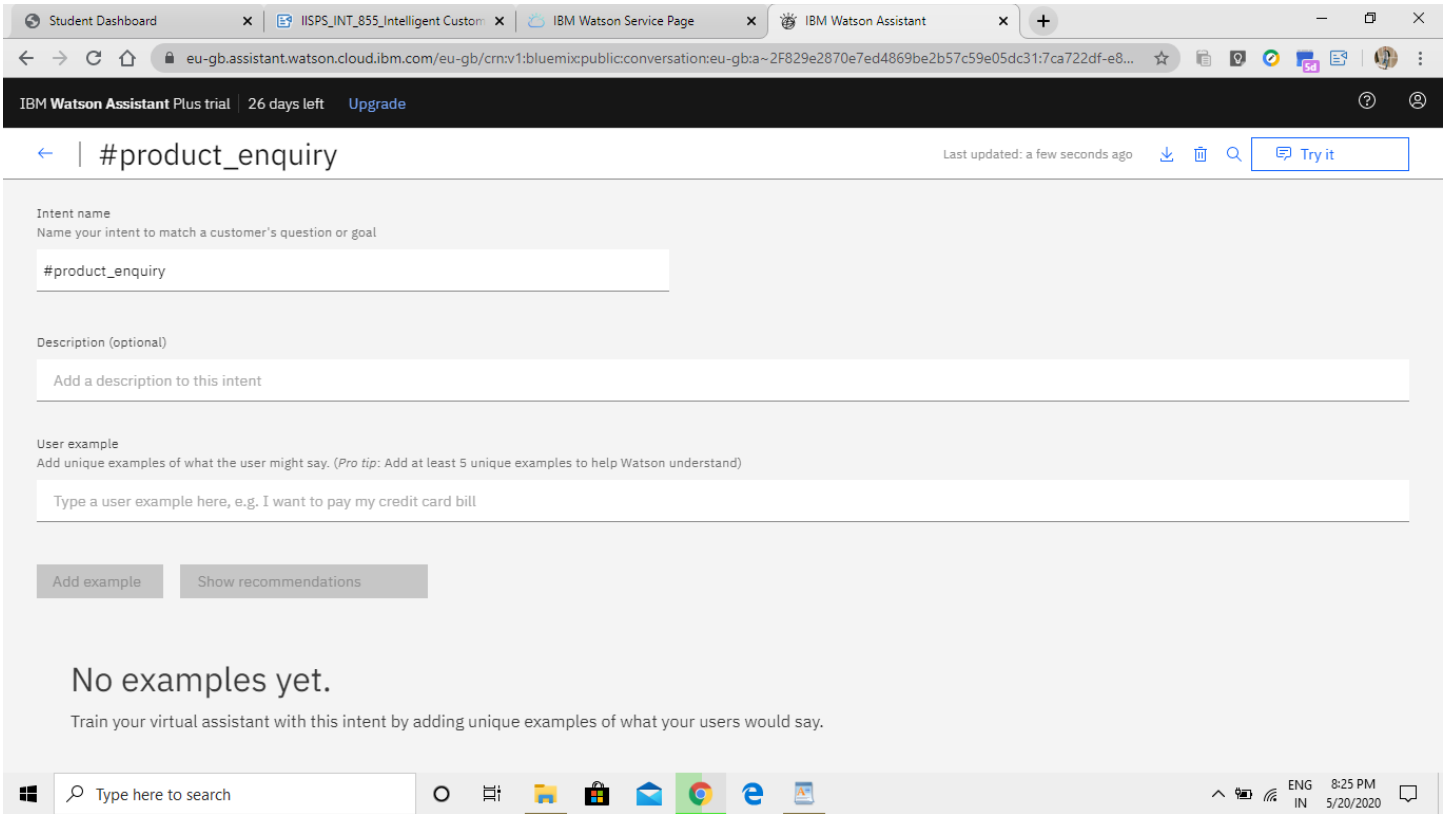
The default customer care dialog does not have a way to deal with any questions involving outside resources, so we will need to add this.

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

From the Customer Care Sample Skill panel, select the Intents tab.

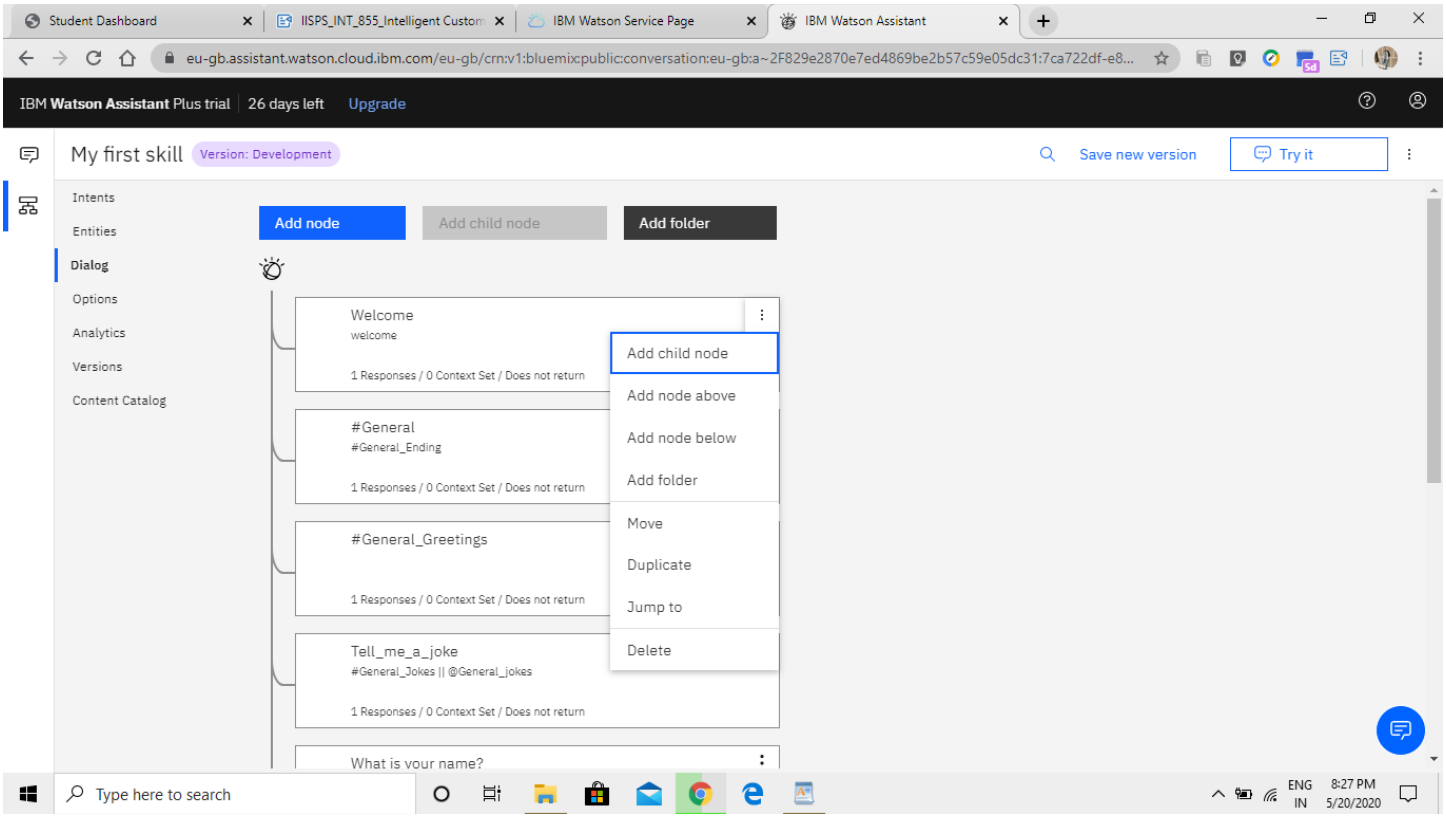
Click the Create intent button.

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

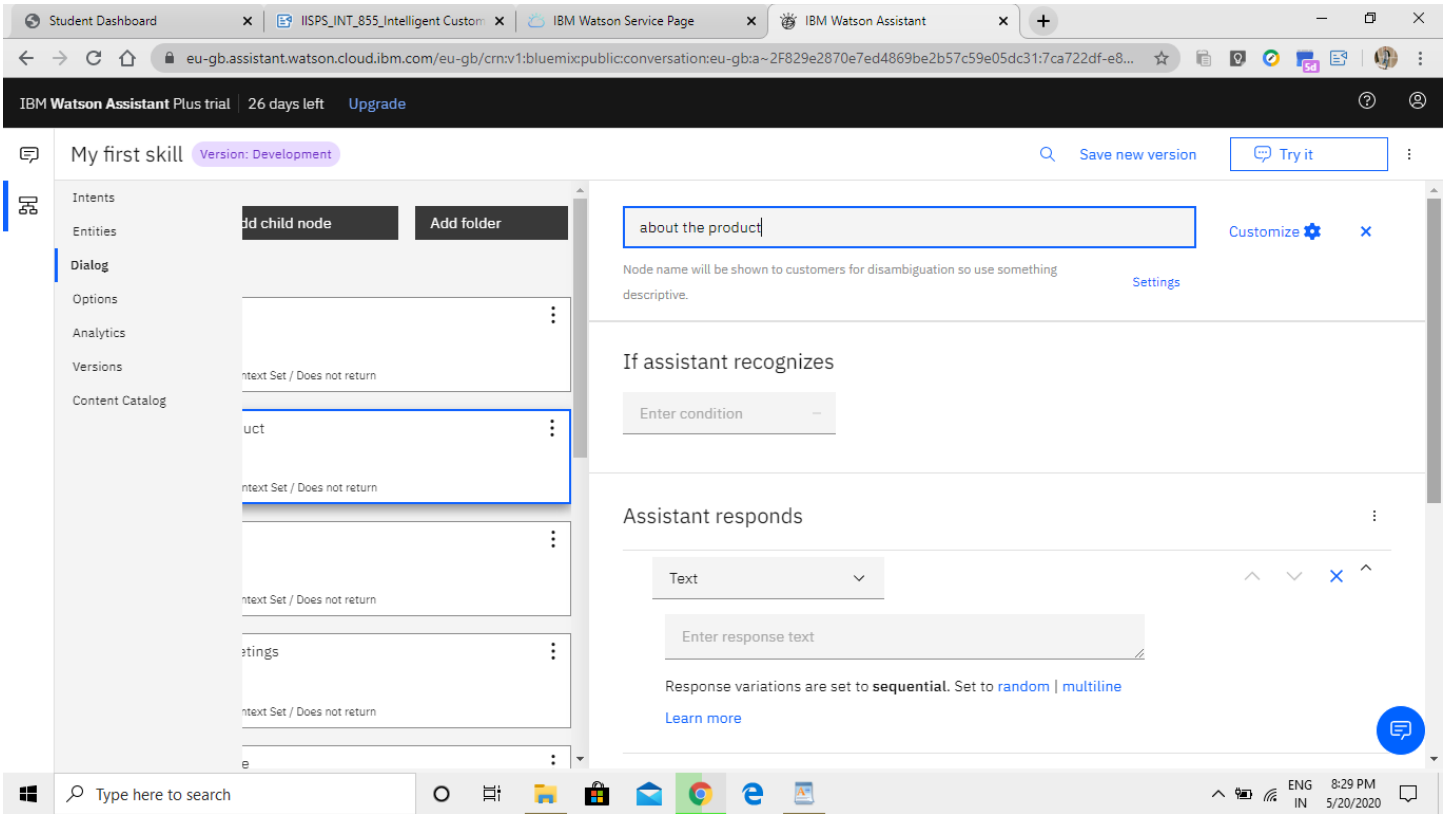


Create new dialog node

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 Small Talk node [2], and select the Add node below [3] option.



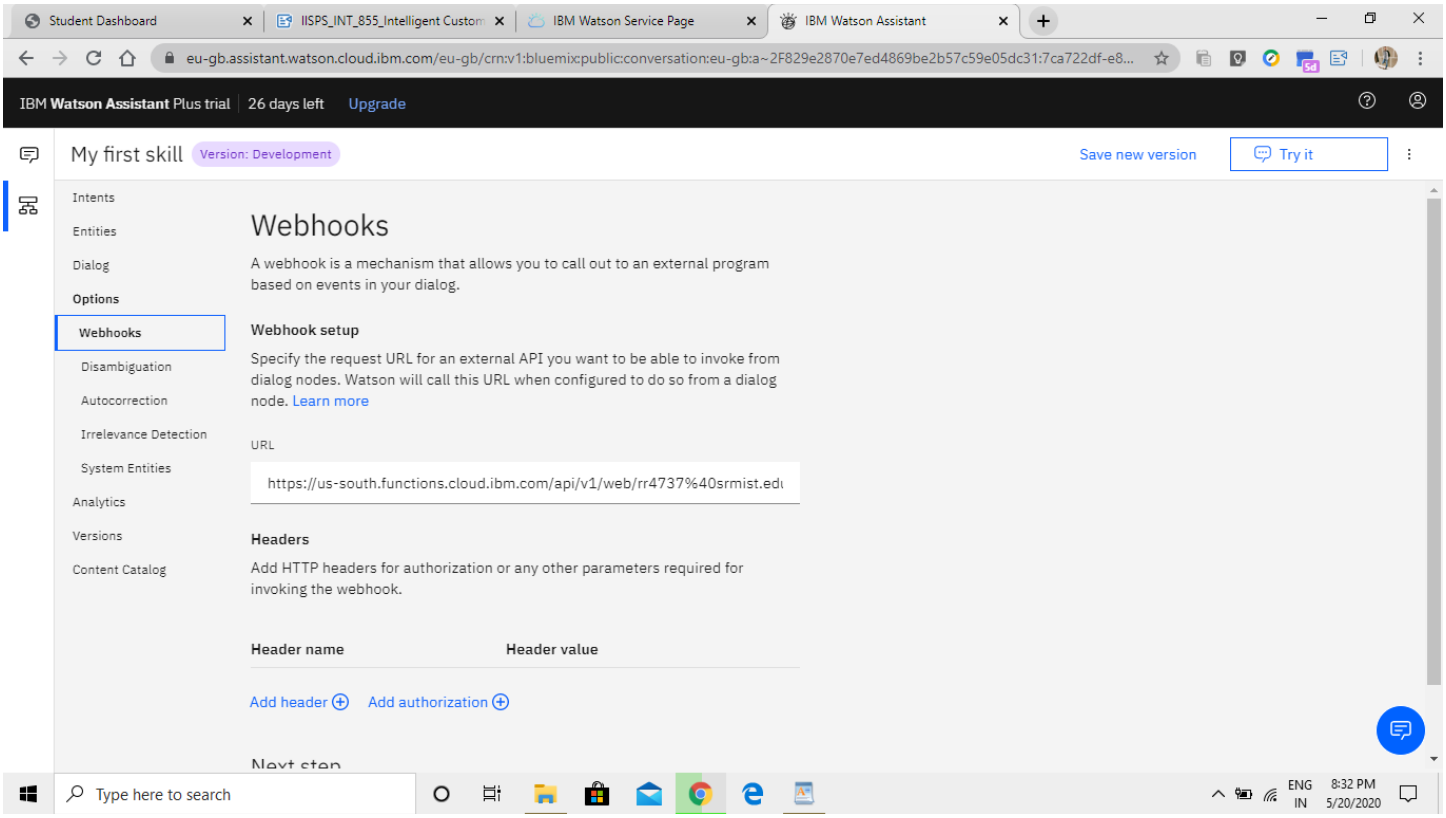
Name the node "Ask about product" [1] and assign it our new intent [2].



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

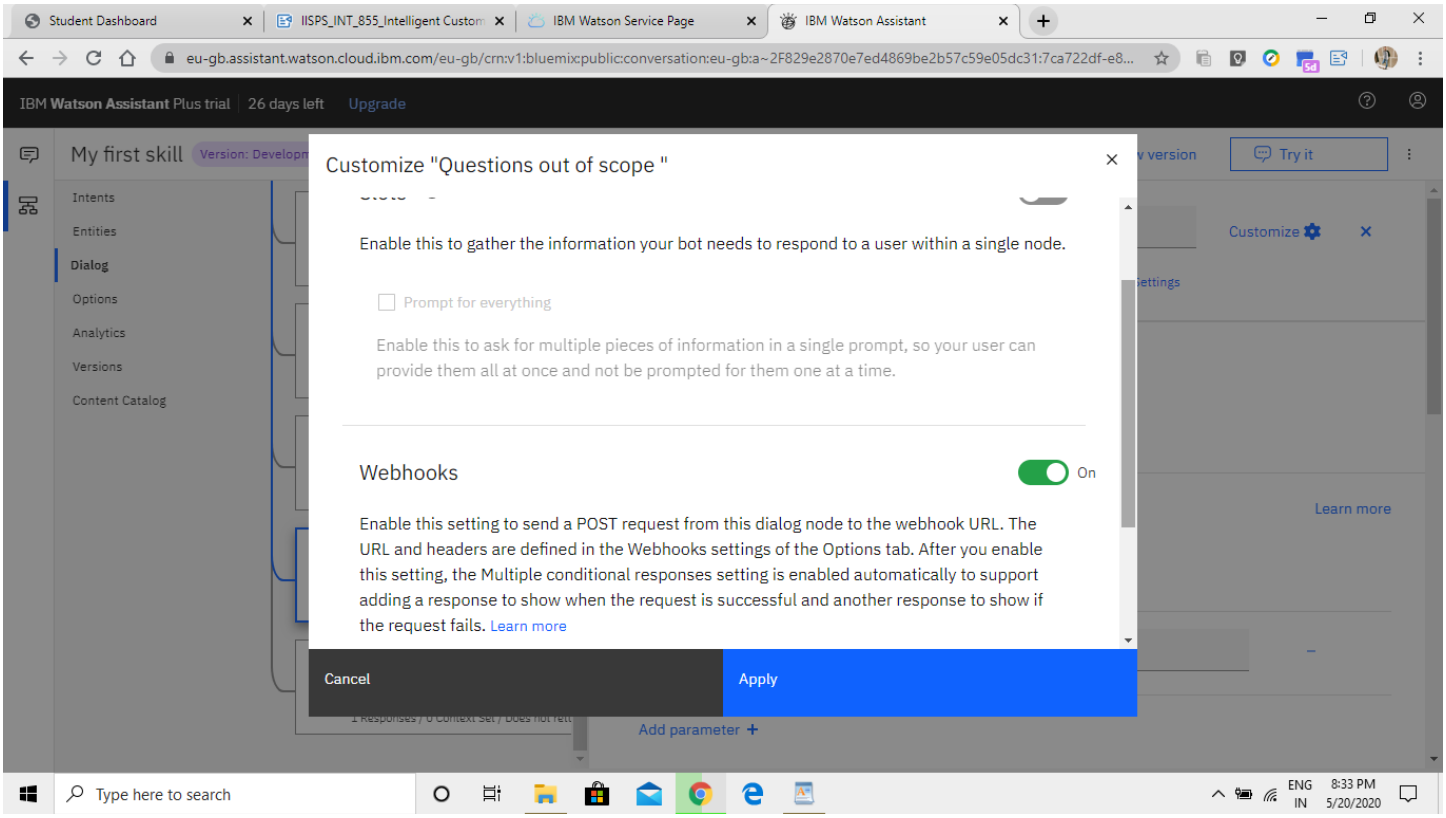
Enable webhook from Assistant

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



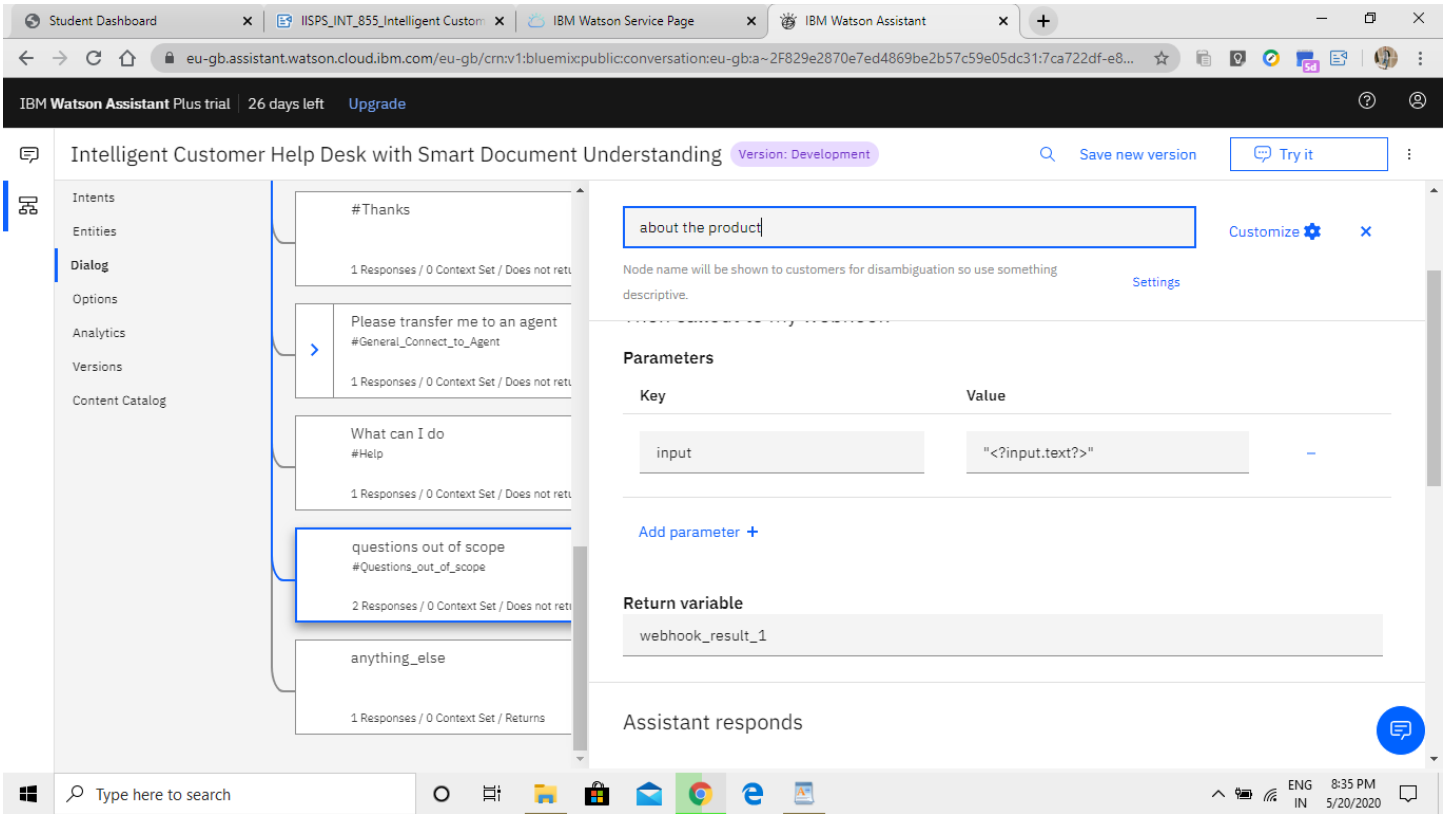
Enter the public URL endpoint for your action [2].

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



Click Apply.

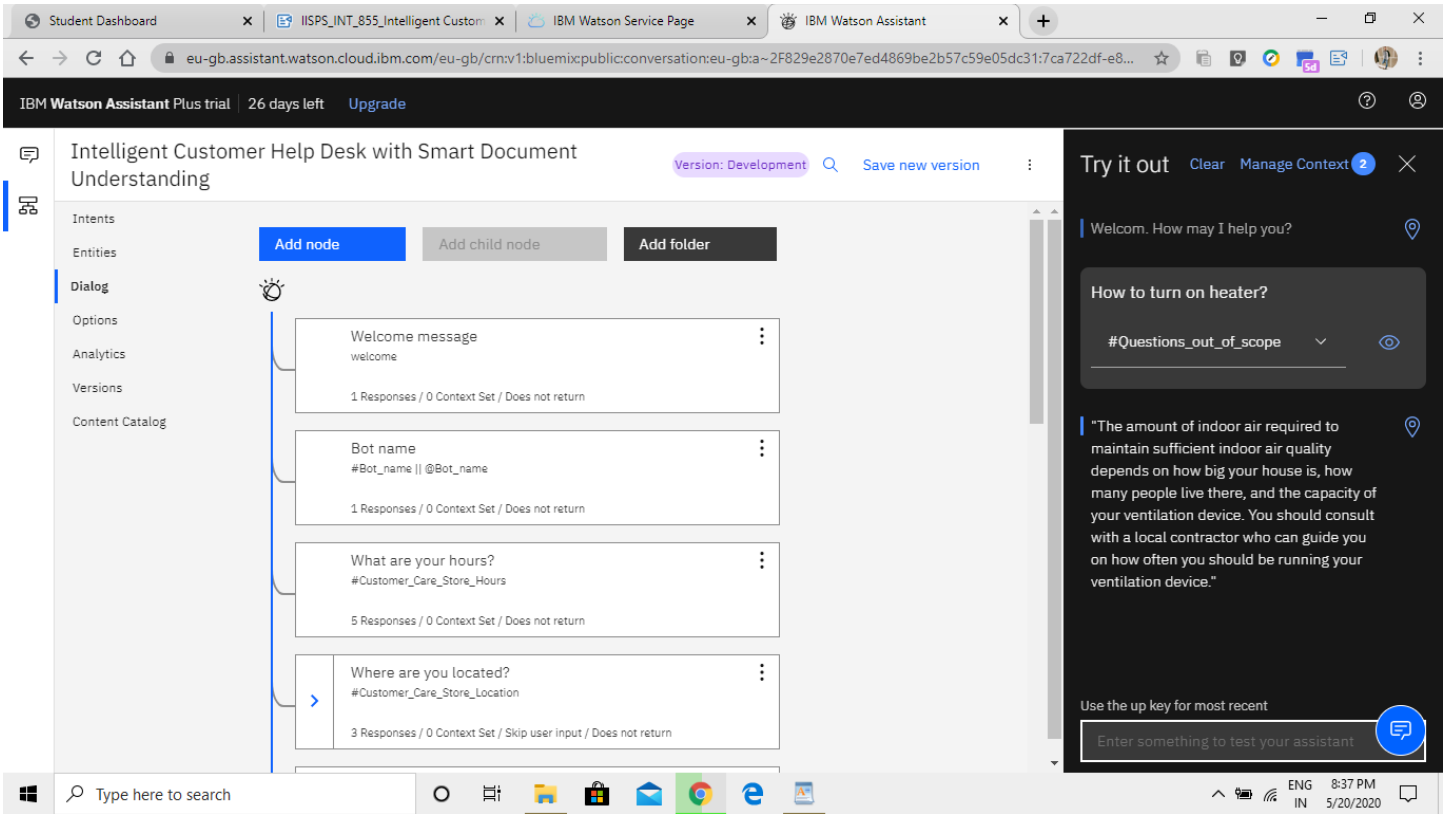
The dialog node should have a Return variable [1] set automatically to $webhook_result_1. This is the variable name you can use to access the result from the Discovery service query.



You 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:
Test in Assistant Tooling

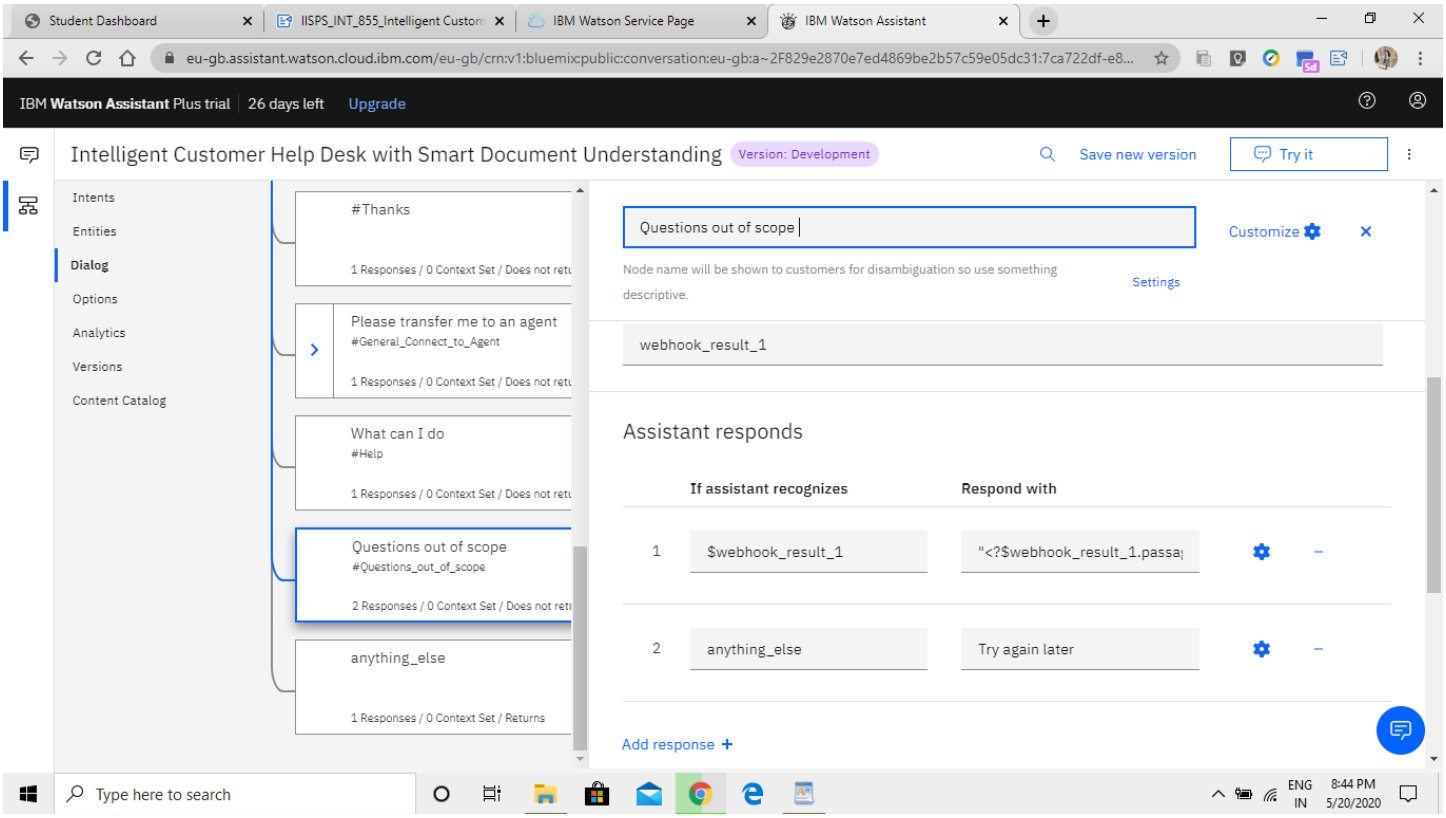
From the Dialog panel, click the Try it button located at the top right side of the panel.
Enter some user input:



Note that the input "how do I turn on the heater?" has triggered our Ask about product 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.

You can also verify that the call was successfully completed by clicking on the Manage Context button at the top right. The response from the Discovery query will be stored in the $webhook_result_1 variable:



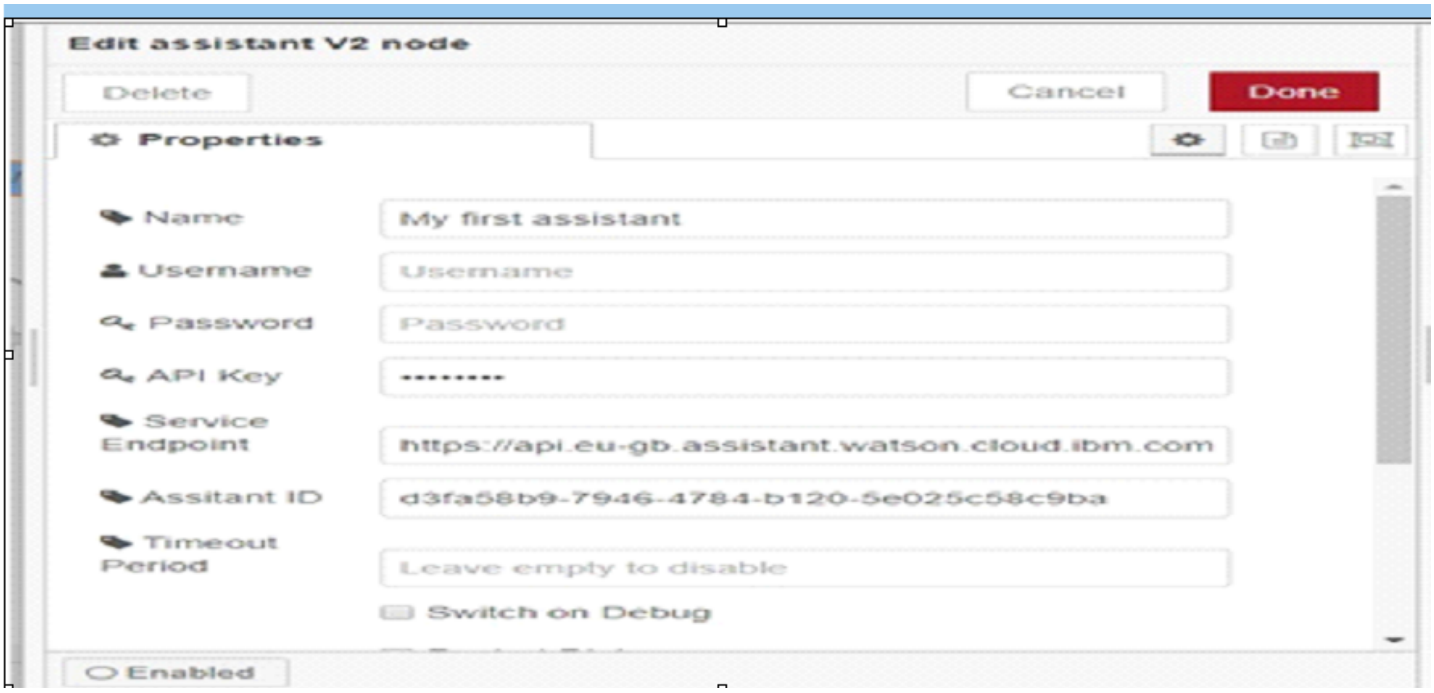
4.Create flow and configure node:

lintegration of Watson assistant in Node-RED:

1. Double-click on the Watson assistant node

2. Give a name to your node and enter the username, password and work space id of

your Watson assistant service.



1.After entering all the information click on Done

2.Drag inject node on to the flow from the Input section

3.Drag Debug on to the flow from the output section

4.Double-click on the inject node

5.Select the payload as a string

6.Enter a sample input to be sent to the assistant service and click on done

7.Connect the nodes as shown below and click on Deploy

8.Open Debug window as shown below

9.Click on the button to send input text to the assistant node

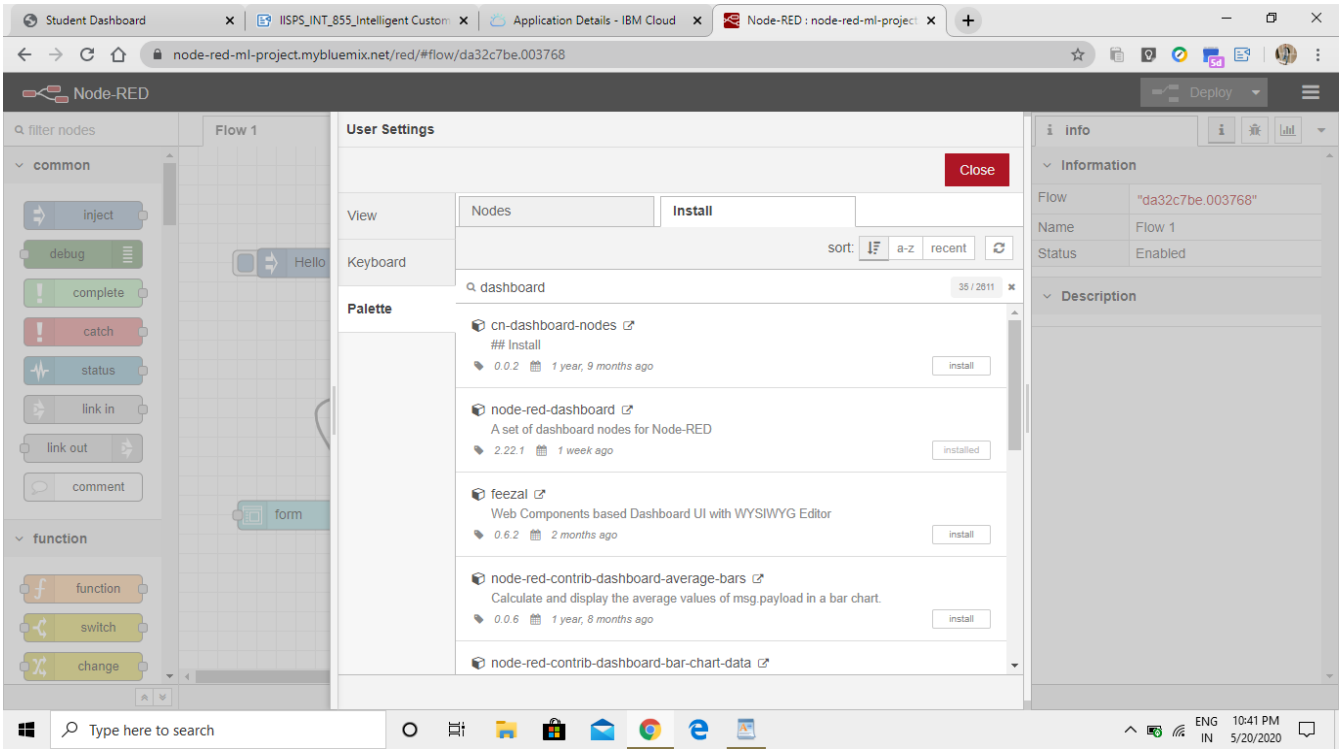
10.Observe the output from the assistant service node

11.The Bot output is located inside “output.text"
12.Drag the function node to parse the JSON data and get the bot response

13.Double click on the function node and enter the JSON parsing code as shown below and click on done

14.Connect the nodes as shown below and click on Deploy

15.Re-inject the flow and observe the parsed output



1.For creating a web application
UI we need “dashboard“ nodes which should be installed manually.

2.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

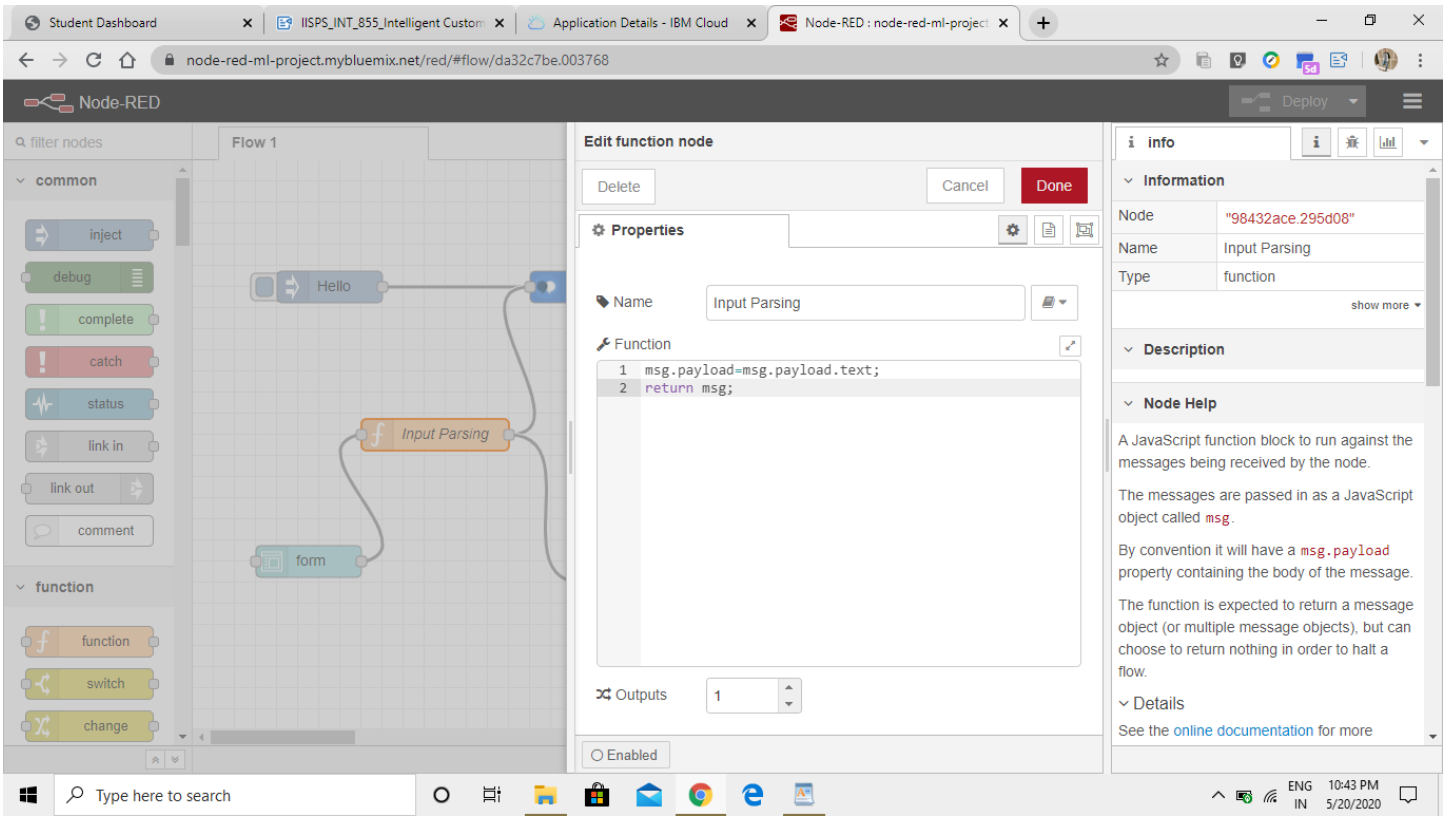
Click on the edit button to add the “Group” name and “Tab”name

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.

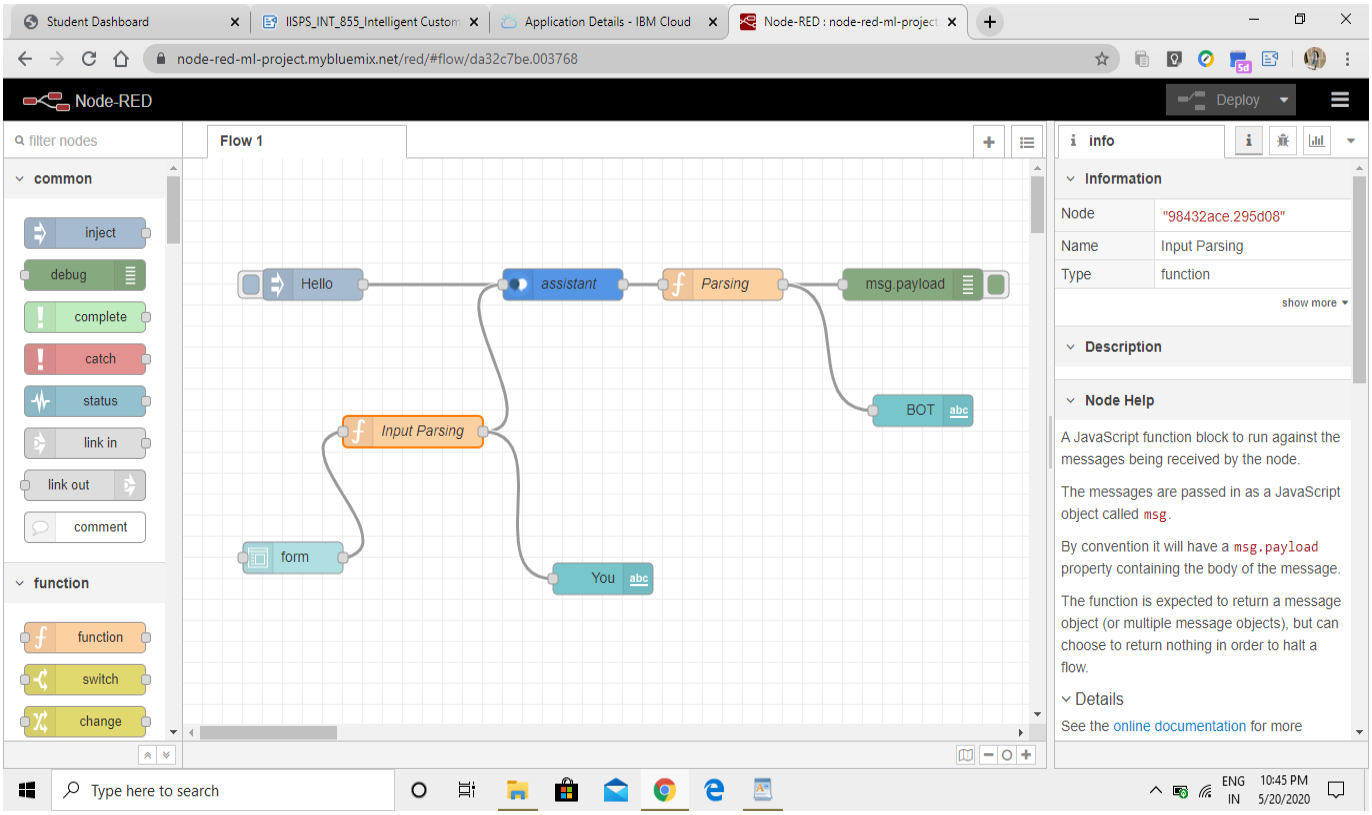


Click on done
Connect the form output to the input of the function node and output of the function to input of assistant node
Search for “text” node from the “dashboard”section
Drag two “text” nodes on to the flow
Double click on the first text node, change the label as “You” and click on Done
Double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

5. FLOW CHART

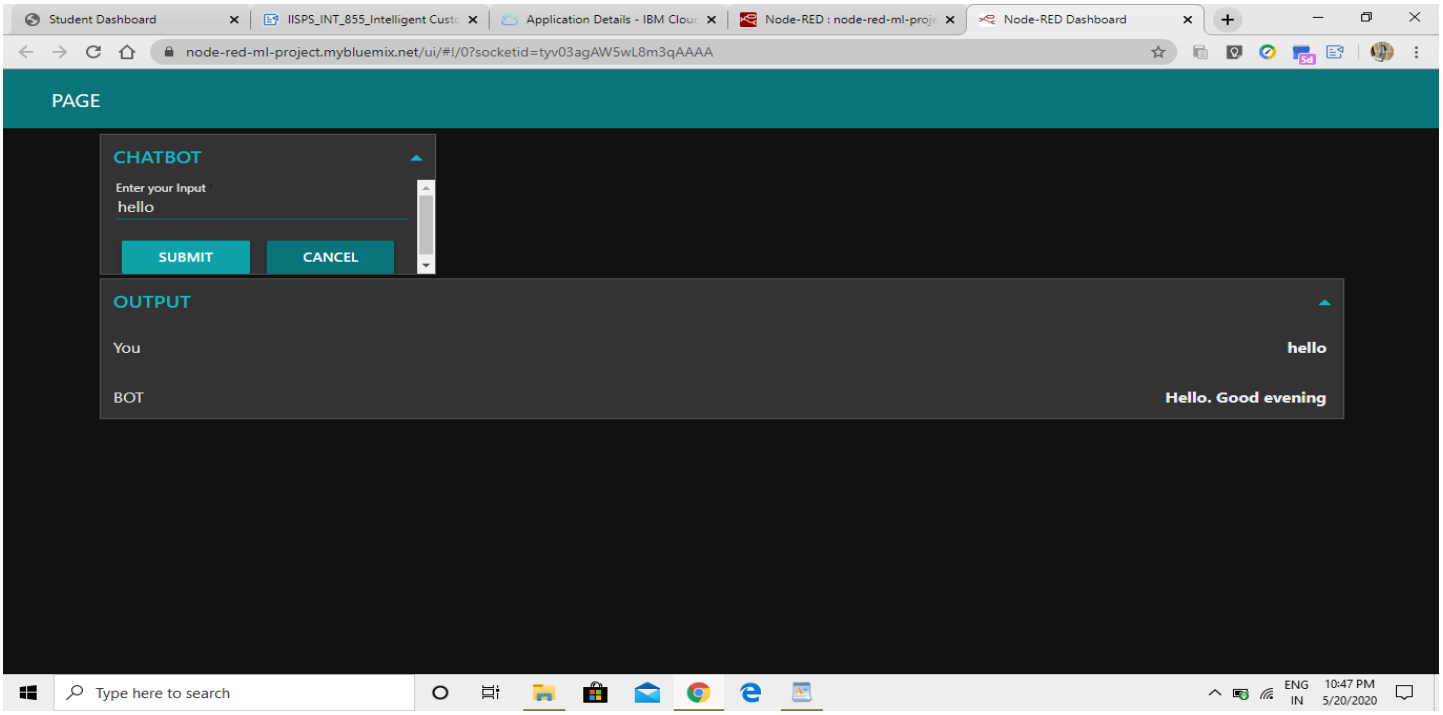
At first go to manage pallete and install dashboard. Now,Create the flow with the help following node:

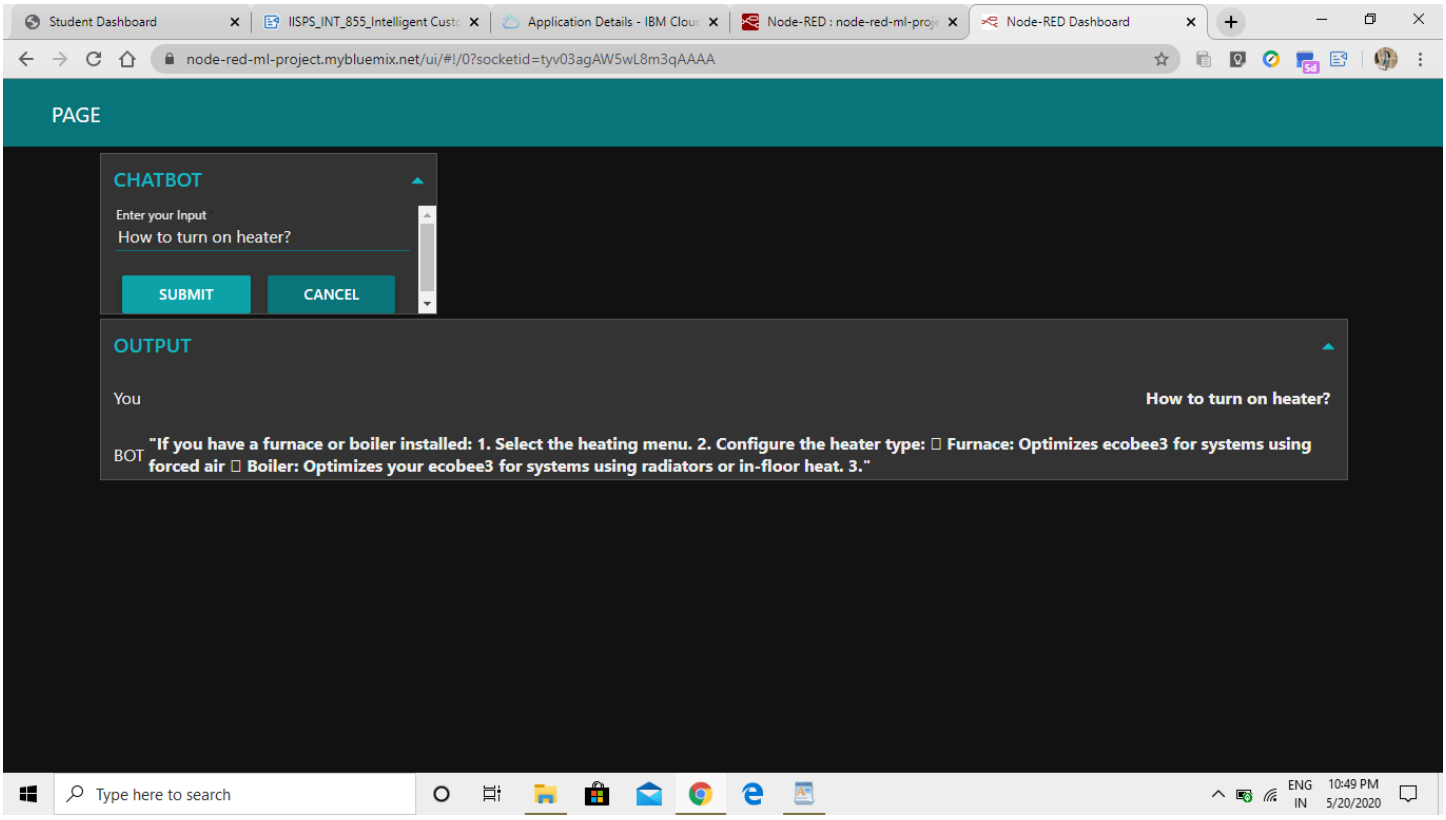
Inject
Assistant
Debug
Function
Ui_Form
Ui_Text



6.RESULTS

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


Student Dashboard
IISPS_INT_855_Intelligent Custo
Application Details - IBM Cloud
Node-RED : node-red-ml-proje
Node-RED Dashboard
node-red-ml-project.mybluemix.net/ui/#!/0?socketid=tyv03agAW5wL8m3qAAAA
PAGE
CHATBOT
Enter your Input
hello
SUBMIT
CANCEL
OUTPUT
You
hello
BOT
Hello. Good evening
Type here to search
ENG
IN
10:47 PM
5/20/2020


Student Dashboard
IISPS_INT_855_Intelligent Custo
Application Details - IBM Cloud
Node-RED : node-red-ml-proje
Node-RED Dashboard
node-red-ml-project.mybluemix.net/ui/#!/0?socketid=tyv03agAW5wL8m3qAAAA
PAGE
CHATBOT
Enter your Input
How to turn on heater?
SUBMIT
CANCEL
OUTPUT
You
How to turn on heater?
BOT
"If you have a furnace or boiler installed: 1. Select the heating menu. 2. Configure the heater type: Furnace: Optimizes ecobee3 for systems using forced air Boiler: Optimizes your ecobee3 for systems using radiators or in-floor heat. 3."
Type here to search
ENG
IN
10:49 PM
5/20/2020

7.ADVANTAGES & DISADVANTAGES

Advantages:

Campanies 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.
can ask queries from home.

Disadvantages:

1. Some times chat bot can mislead customers

2.Giving same answer for different sentiments.

3.Some times cannot connect to customer sentiments and intentions.

8.APPLICATIONS

It can deploy in popular social media applications like facebook,slack,telegram.
Chatbot can deploy any website to clarify basic doubts of viewers.
chat bot is the easy way of communication
It can be used for various applications
it is used to know about the product
it can also be used in knowing basic information

9.CONCLUSION

By doing the above procedure and all we successfully created Intelligent help desk smart chat bot using Watson assistant, Watson discovery, Node-RED and cloud-functions.
The conclusion for this is that it is the easy way of communication where we can resolve the

customer or the consumer queries without the help of the person we can do it virtually.
So customer can access 24 hours for the queries what he have and get resolved his problem simply by sitting in his place and asking the bot some questions.
If the bot does not understand it just give you the number of the customer care where you can contact them for future queries.

10.FUTURE SCOPE

We can include Watson studio text to speech and speech to text services to access the chat bot hands free. This is one of the future scope of this project.

11. BIBILOGRAPHY

we used some IBM guidelines etc to resolve our problems.

APPENDIX

Source Code

CloudFunction(Node.js)

```
/**
*
        @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 thisaction

*

        @paramCloudFunctionsactionsacceptasingleparameter,whichmustbeaJSONobject.
```

```
*

        @return The output of this action, which must be a JSONobject.

*

*/

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': '2020-05-09'

});

}

else {

discovery = new DiscoveryV1({ 'username': params.username, 'password': params.password, 'url':
params.url,
'version': '2020-05-11'

});

}
```

```
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);

});

});

}
```

2.Node Red(flow.json)

```
 [
   {
      "id": "d0d85ff2.0f31e",
      "type": "tab",
      "label": "Flow 1",
      "disabled": false,
      "info": ""
```

```
},
{
    "id": "81a35420.f26278",
    "type": "ui_form",
    "z": "d0d85ff2.0f31e",
    "name": "form",
    "label": "",
    "group": "eee5eb1.9de6f18",
    "order": 0,
    "width": "6",
    "height": "6",
    "options": [
        {
            "label": "enter your queries",
            "value": "text",
            "type": "text",
            "required": true,
            "rows": null
        }
    ],
    "formValue": {
        "text": ""
    },
    "payload": "",
    "submit": "submit",
    "cancel": "cancel",
    "topic": "",
    "x": 70,
    "y": 400,
    "wires": [
        [
            "2fc109df.9935e6"
        ]
    ]
},
```

```
{
    "id": "e4411f54.83b62",
    "type": "ui_text",
    "z": "d0d85ff2.0f31e",
    "group": "eee5eb1.9de6f18",
    "order": 1,
    "width": 0,
    "height": 0,
    "name": "",
    "label": "you",
    "format": "{{msg.payload}}",
    "layout": "row-spread",
    "x": 310,
    "y": 400,
    "wires": []
},
{
    "id": "8ea21b5b.9dfc78",
    "type": "inject",
    "z": "d0d85ff2.0f31e",
    "name": "",
    "topic": "",
    "payload": "hello",
    "payloadType": "str",
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "x": 90,
    "y": 120,
    "wires": [
        [
            "3464ad27.b91b72"
        ]
    ]
```

```
},
{
    "id": "2fc109df.9935e6",
    "type": "function",
    "z": "d0d85ff2.0f31e",
    "name": "input parse",
    "func": "msg.payload=msg.payload.text;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 190,
    "y": 300,
    "wires": [
        [
            "3464ad27.b91b72",
            "e4411f54.83b62"
        ]
    ]
},
{
    "id": "971684ee.01b3d8",
    "type": "function",
    "z": "d0d85ff2.0f31e",
    "name": "parsing",
    "func": "msg.payload=msg.payload.output.text[0];\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 560,
    "y": 200,
    "wires": [
        [
            "ae891913.e5e228",
            "de71deca.ca4b1"
        ]
    ]
},
```

```
{
    "id": "ae891913.e5e228",
    "type": "ui_text",
    "z": "d0d85ff2.0f31e",
    "group": "eee5eb1.9de6f18",
    "order": 0,
    "width": "6",
    "height": "9",
    "name": "",
    "label": "bot",
    "format": "{{msg.payload}}",
    "layout": "row-left",
    "x": 730,
    "y": 320,
    "wires": []
},
{
    "id": "de71deca.ca4b1",
    "type": "debug",
    "z": "d0d85ff2.0f31e",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "x": 730,
    "y": 80,
    "wires": []
},
{
    "id": "3464ad27.b91b72",
    "type": "watson-conversation-v1",
    "z": "d0d85ff2.0f31e",
```

```
    "name": "customer service bot",
    "workspaceid": "2b5beca9-21e4-4391-b502-309530f12a63",
    "multiuser": false,
    "context": true,
    "empty-payload": false,
    "service-endpoint": "",
    "timeout": "",
    "optout-learning": false,
    "x": 340,
    "y": 220,
    "wires": [
      [
        "971684ee.01b3d8"
      ]
    ]
  },
  {
    "id": "eee5eb1.9de6f18",
    "type": "ui_group",
    "z": "",
    "name": "form",
    "tab": "59bfe54e.277f4c",
    "order": 2,
    "disp": true,
    "width": "6",
    "collapse": false
  },
  {
    "id": "59bfe54e.277f4c",
    "type": "ui_tab",
    "z": "",
    "name": "Home",
    "icon": "dashboard",
    "disabled": false,
    "hidden": false
```

```
    }
]


[
    {
        "id": "d0d85ff2.0f31e",
        "type": "tab",
        "label": "Flow 1",
        "disabled": false,
        "info": ""
    },
    {
        "id": "eee5eb1.9de6f18",
        "type": "ui_group",
        "z": "",
        "name": "form",
        "tab": "59bfe54e.277f4c",
        "order": 2,
        "disp": true,
        "width": "6",
        "collapse": false
    },
    {
        "id": "59bfe54e.277f4c",
        "type": "ui_tab",
        "z": "",
        "name": "Home",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    },
    {
        "id": "22f35ab6.d3ff06",
        "type": "ui_group",
```

```
    "z": "",
    "name": "Default",
    "tab": "59bfe54e.277f4c",
    "order": 1,
    "disp": true,
    "width": "6",
    "collapse": false
  },
  {
    "id": "3b2ea5a6.df5fca",
    "type": "ui_base",
    "theme": {
      "name": "theme-light",
      "lightTheme": {
        "default": "#0094CE",
        "baseColor": "#0094CE",
        "baseFont": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",
        "edited": true,
        "reset": false
      },
      "darkTheme": {
        "default": "#097479",
        "baseColor": "#097479",
        "baseFont": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",
        "edited": false
      },
      "customTheme": {
        "name": "Untitled Theme 1",
        "default": "#4B7930",
        "baseColor": "#4B7930",
        "baseFont": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"
      },
```

```
"themeState": {
    "base-color": {
        "default": "#0094CE",
        "value": "#0094CE",
        "edited": false
    },
    "page-titlebar-backgroundColor": {
        "value": "#0094CE",
        "edited": false
    },
    "page-backgroundColor": {
        "value": "#fafafa",
        "edited": false
    },
    "page-sidebar-backgroundColor": {
        "value": "#ffffff",
        "edited": false
    },
    "group-textColor": {
        "value": "#1bbfff",
        "edited": false
    },
    "group-borderColor": {
        "value": "#ffffff",
        "edited": false
    },
    "group-backgroundColor": {
        "value": "#ffffff",
        "edited": false
    },
    "widget-textColor": {
        "value": "#111111",
        "edited": false
    },
    "widget-backgroundColor": {
```

```
            "value": "#0094ce",
            "edited": false
          },
          "widget-borderColor": {
            "value": "#ffffff",
            "edited": false
          },
          "base-font": {
            "value": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"
          }
        },
        "angularTheme": {
          "primary": "indigo",
          "accents": "blue",
          "warn": "red",
          "background": "grey"
        }
      },
      "site": {
        "name": "Node-RED Dashboard",
        "hideToolbar": "false",
        "allowSwipe": "false",
        "lockMenu": "false",
        "allowTempTheme": "true",
        "dateFormat": "DD/MM/YYYY",
        "sizes": {
          "sx": 48,
          "sy": 48,
          "gx": 6,
          "gy": 6,
          "cx": 6,
          "cy": 6,
          "px": 0,
          "py": 0
```

```
        }
    }
},
{
    "id": "81a35420.f26278",
    "type": "ui_form",
    "z": "d0d85ff2.0f31e",
    "name": "form",
    "label": "",
    "group": "eee5eb1.9de6f18",
    "order": 0,
    "width": "6",
    "height": "6",
    "options": [
        {
            "label": "enter your queries",
            "value": "text",
            "type": "text",
            "required": true,
            "rows": null
        }
    ],
    "formValue": {
        "text": ""
    },
    "payload": "",
    "submit": "submit",
    "cancel": "cancel",
    "topic": "",
    "x": 70,
    "y": 400,
    "wires": [
        [
            "2fc109df.9935e6"
        ]
```

```
    ]
},
{
    "id": "e4411f54.83b62",
    "type": "ui_text",
    "z": "d0d85ff2.0f31e",
    "group": "eee5eb1.9de6f18",
    "order": 1,
    "width": 0,
    "height": 0,
    "name": "",
    "label": "you",
    "format": "{{msg.payload}}",
    "layout": "row-spread",
    "x": 310,
    "y": 400,
    "wires": []
},
{
    "id": "8ea21b5b.9dfc78",
    "type": "inject",
    "z": "d0d85ff2.0f31e",
    "name": "",
    "topic": "",
    "payload": "hello",
    "payloadType": "str",
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "x": 90,
    "y": 120,
    "wires": [
        [
            "3464ad27.b91b72"
```

```
        ]
    ]
},
{
    "id": "2fc109df.9935e6",
    "type": "function",
    "z": "d0d85ff2.0f31e",
    "name": "input parse",
    "func": "msg.payload=msg.payload.text;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 190,
    "y": 300,
    "wires": [
        [
            "3464ad27.b91b72",
            "e4411f54.83b62"
        ]
    ]
},
{
    "id": "971684ee.01b3d8",
    "type": "function",
    "z": "d0d85ff2.0f31e",
    "name": "parsing",
    "func": "msg.payload=msg.payload.output.text[0];\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 560,
    "y": 200,
    "wires": [
        [
            "ae891913.e5e228",
            "de71deca.ca4b1"
        ]
```

```
    ]
},
{
    "id": "ae891913.e5e228",
    "type": "ui_text",
    "z": "d0d85ff2.0f31e",
    "group": "eee5eb1.9de6f18",
    "order": 0,
    "width": "6",
    "height": "9",
    "name": "",
    "label": "bot",
    "format": "{{msg.payload}}",
    "layout": "row-left",
    "x": 730,
    "y": 320,
    "wires": []
},
{
    "id": "de71deca.ca4b1",
    "type": "debug",
    "z": "d0d85ff2.0f31e",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "x": 730,
    "y": 80,
    "wires": []
},
{
    "id": "3464ad27.b91b72",
```

```
        "type": "watson-conversation-v1",
        "z": "d0d85ff2.0f31e",
        "name": "customer service bot",
        "workspaceid": "2b5beca9-21e4-4391-b502-309530f12a63",
        "multiuser": false,
        "context": true,
        "empty-payload": false,
        "service-endpoint": "",
        "timeout": "",
        "optout-learning": false,
        "x": 340,
        "y": 220,
        "wires": [
            [
                "971684ee.01b3d8"
            ]
        ]
    }
]
```


References:

https://www.ibm.com/cloud/architecture/tutorials/cognitive_discovery

h:ttps//cloud.ibm.com/docs/assistant?topic=assistant-getting-started

h ttps//developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chatbot-on-nodered/

h:ttps//www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red

h:ttps//github.com/IBM/watson-discovery-sdu-with-assistant

h: ttps//www.youtube.com/watch?v=Jpr3wVH3FVA