PROJECT REPORT

Intelligent Customer Help Desk with Smart Document Understanding - SB52650

Name**: Manish Anand Pawar**

E-mail **: heymanishp@gmail.com**

Category**: Artificial Intelligence Internship at SMARTINTERNZ**

Application ID**: SPS_APL_20200003541**

Project ID**: SPS_PRO_99**

INDEX

1. INTRODUCTION

1.1 Overview:

A customer care chat bot can answer simple questions, such as store locations and hours, directions, and perhaps even making appointments. In this project, the queries will be handled in a better way. If the customer's question is about the operation of a device, the application shall pass the question onto Watson Discovery Service and we can handle the queries in a better way. We will build a chat bot that uses various Watson AI Services like Watson Discovery, Watson Assistant, Watson Cloud functions and Node-Red and deliver an effective user friendly Web User Interface.

- Project needs:
  IBM Cloud, IBM Watson, Node-Red, NodeJS
- Functional needs:
  IBM Cloud
- Technical needs:
  Artificial Intelligence, Machine Learning, Watson AI, NodeJS
- Software needs:
  Watson Assistant, Watson Discovery, Node-Red
- Project Deliverables:
  Intelligent Chatbot with Smart document understand
- Project Team:
  Manish Pawar

1.2 Purpose:

The typical customer care chat bot can answer simple questions, such as store locations and hours, directions, and even making appointments.When a question falls outside of the scope of the predetermined question set, the option is typically to tell the customer the question is not valid or offer to speak to a real person. In this project, there will be an 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.

1.2.1 Scope of Work

- Create a customer care dialog skill in Watson Assistant.
- Use Smart Document Understanding to build an enhanced Watson Discovery collection.
- Create an IBM Cloud Functions web action that allows Watson Assistant to post queries to Watson Discovery.
- Build a web application with integration to all these services and deploy the same on IBM Cloud Platform.

2. LITERATURE SURVEY

2.1 Existing problem:

Generally Chatbots means getting input from users and getting only response questions and for some questions the output from bot will be like “try again”, “I don’t understand”, “will you repeat again”, and so on… and directs customer to customer agent but a good customer Chatbot 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 chatbots.

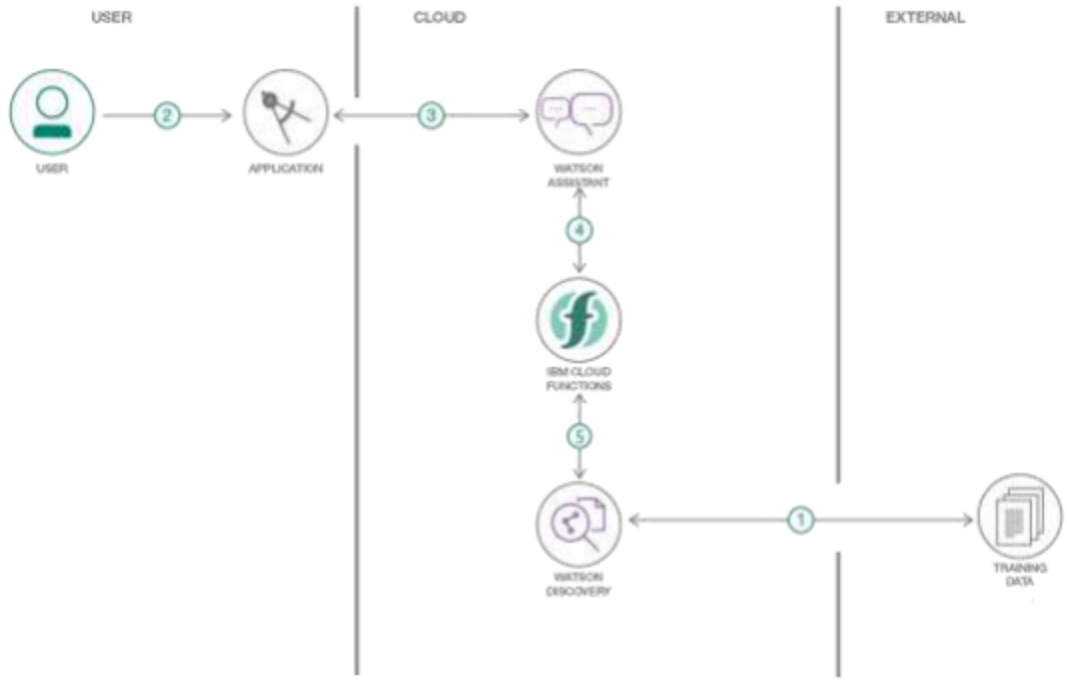
2.2 Proposed solution:

For the above problem to get solved we have to put a virtual agent in chat bot so that it can understand the queries that are posted by customers. The virtual agent should be trained on some modules of records based on the company background so that it can answer the queries related to the product or related to company. In this project I have used Watson Discovery to achieve the above solution,along with Watson Assistant and built an User Interface using Node-RED.

3. THEORETICAL ANALYSIS

3.1 Architecture/flow

The following flow is the basic working flow of the project.



3.2 Hardware/Software design:

- Create IBM Cloud services
- Configure Watson Discovery
- Create IBM Cloud Functions action
- Configure Watson Assistant
- Create flow and configure node
- Deploy and run Node Red app.

4. EXPERIMENTAL INVESTIGATIONS

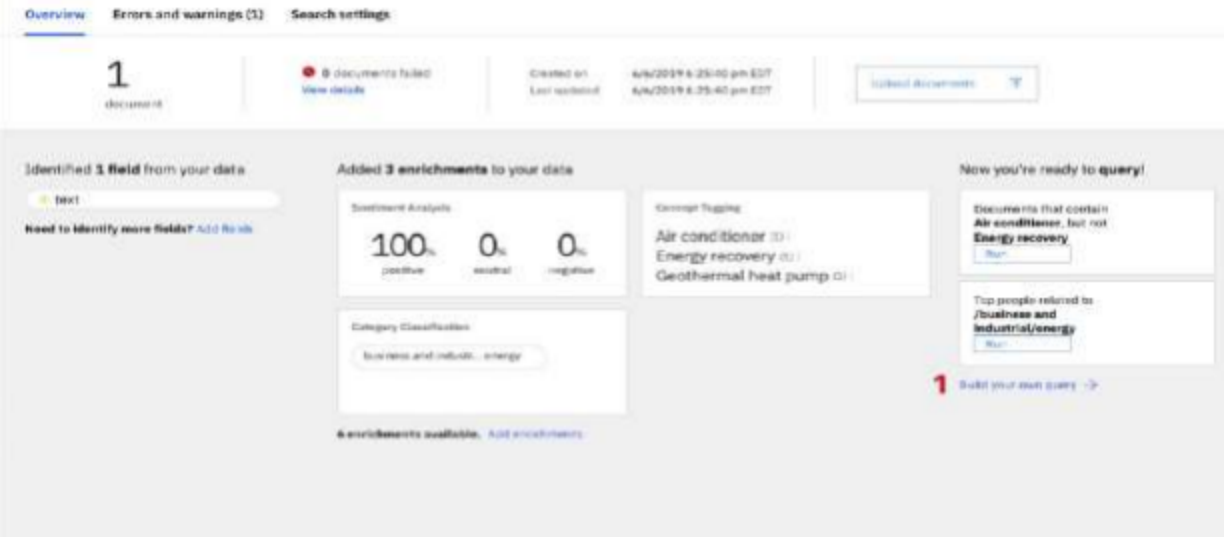
4.1. Create IBM Cloud services:

- Watson Discovery
- Watson Assistant
- Node Red

4.2. Configure Watson Discovery

1. Import the document
2. Launch the Watson Discovery tool and create a new data collection by selecting the Upload your own data option.
3. Give the data collection a unique name.
4. When prompted, select and upload the ecobee3_UserGuide.pdf file located in the data directory of your local repository.
5. The Ecobee is a popular residential thermostat that has a wifi interface and multiple configuration options.
6. 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.
7. Enter queries related to the operation of the thermostat and view the results.
8. Annotate with SDU. From the Discovery collection panel, click the Configure data button (located in the top right corner) to start the SDU process.

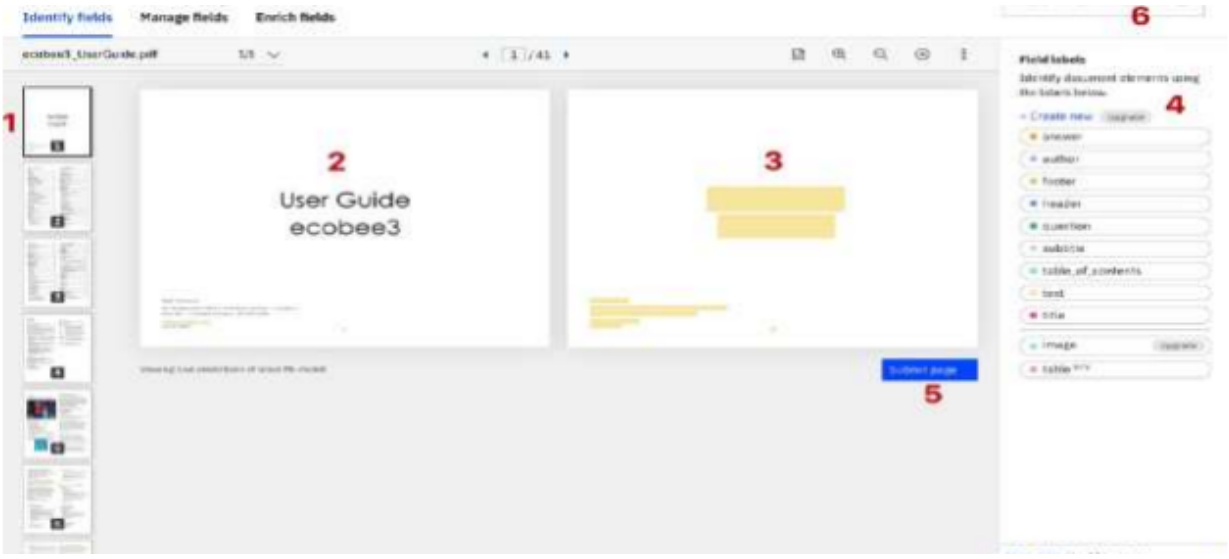
9. Here is the layout of the Identify fields tab of the SDU annotation panel:



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.

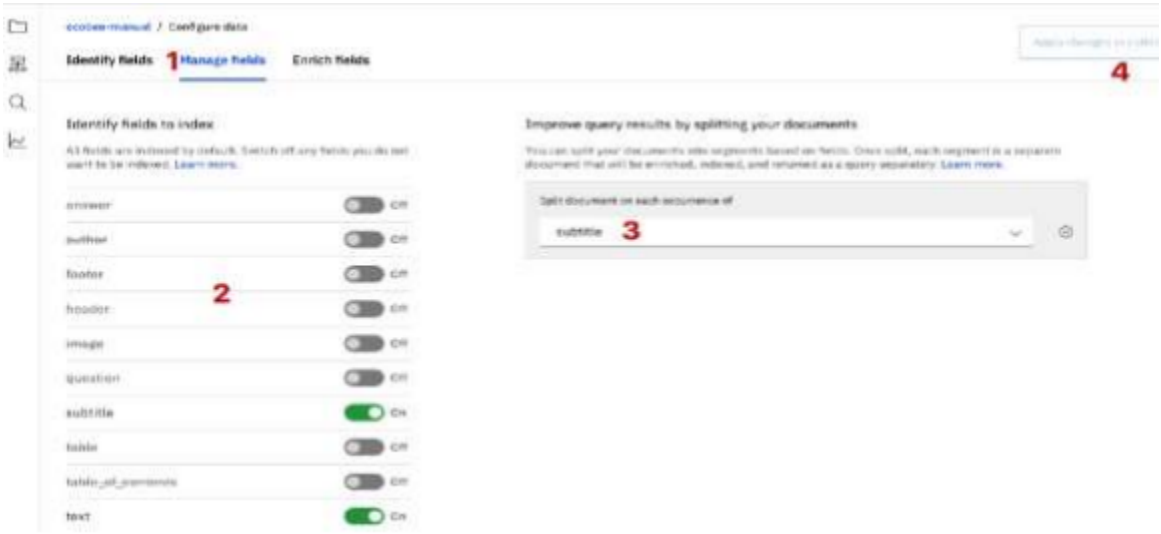
So we do:

- [1] is the list of pages in the manual. As each is processed, a green check mark will appear on the page.
- [2] is the current page being annotated.
- [3] is where you select text and assign it a label.
- [4] is the list of labels you can assign to the page text.
- Click [5] to submit the page to Discovery.
- Click [6] when you have completed the annotation process.



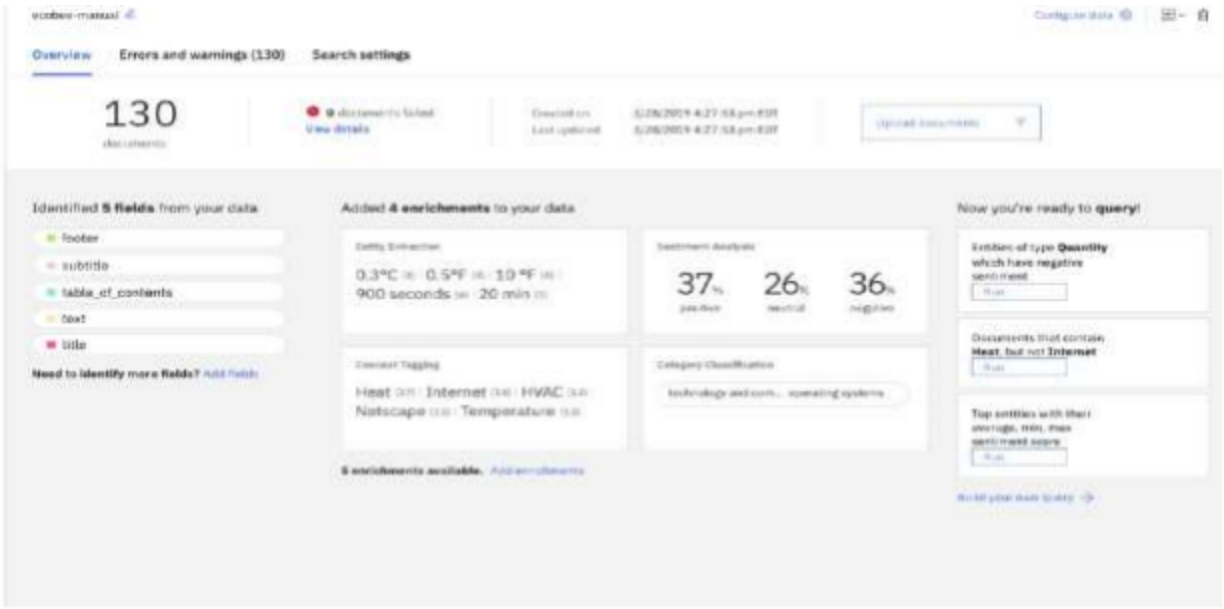
- As you go though the annotations one page at a time, Discovery is learning and should start automatically updating the upcoming pages. Once you get to a page that is already correctly annotated, you can stop, or simply click Submit [5] to acknowledge it is correct. The more pages you annotate, the 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 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 we click the Apply changes to collection button [6], we will be asked to reload the document. We choose the same owner's manual .pdf document as before.

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



- [2] Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except subtitles and text.
- [3] 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, because of splitting the document apart, your collection will look very different.

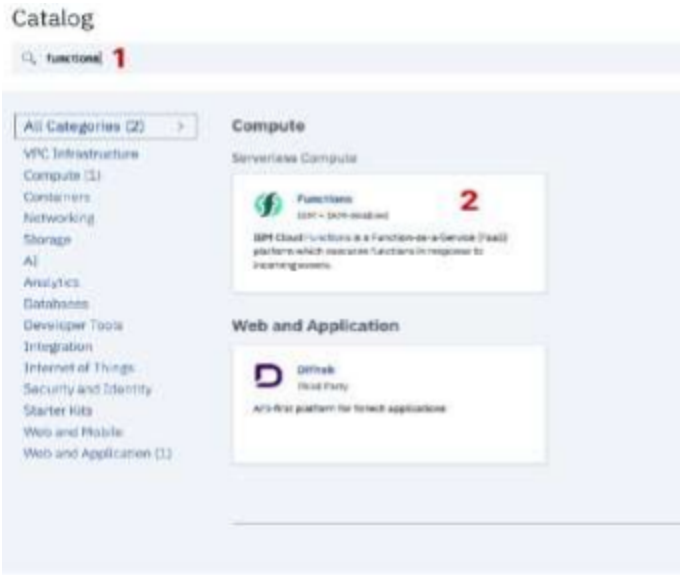
The results to the queries related to the product are very accurate after annotation of the document.



4.3 Create IBM Cloud Functions

Now let us 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]

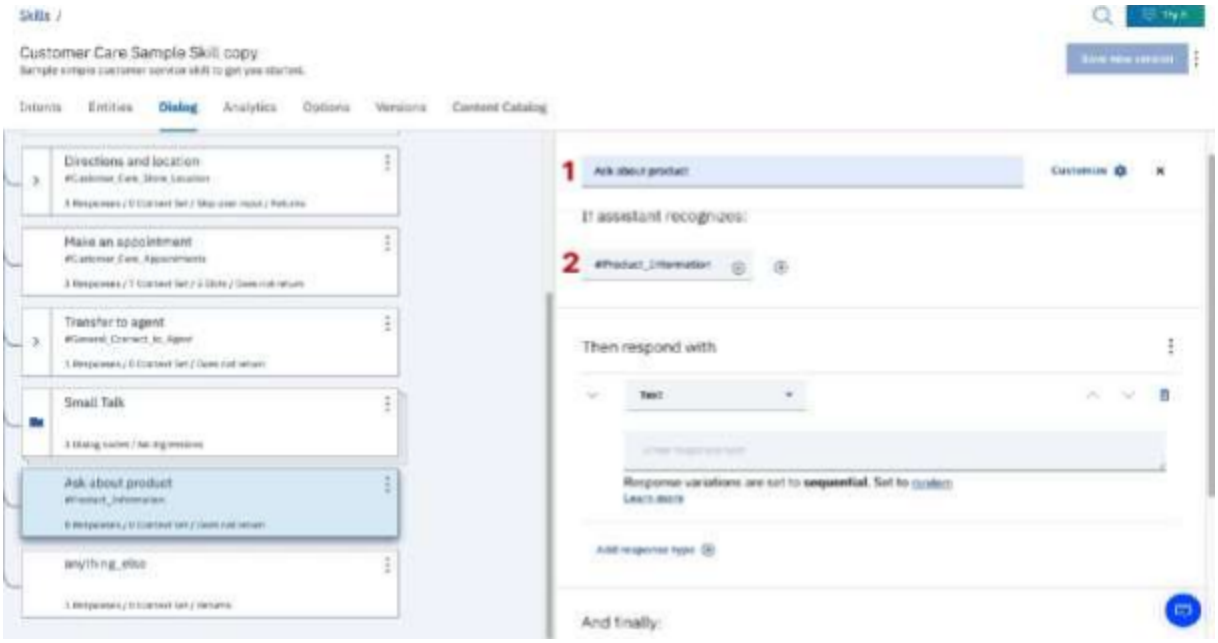


- From the Functions main panel, click on the Actions tab. Then click on Create.
- From the Create panel, select the Create Action option.
- On the Create Action panel, provide a unique Action Name [1], keep the default package [2], and select the Node.js 10 [3] runtime. Click the Create button [4] to create the action.
- Once your action is created, click on the Code tab [1]
  .In the code editor window [2], cut and paste in the code from the cloud_function.js file found in the actions directory of your local repository. The code is pretty straight-forward - it simply connects to the Discovery service, makes a query against the collection, then returns the response.

- Set credentials for your service
- For values, please use the values associated with the Discovery service you created in the previous step.
- Now that the credentials are set, return to the Code panel and press the Invoke button again.
- Now you should see actual results returned from the Discovery service:
    - Next, go to the Endpoints panel
    - Click the checkbox for Enable as Web Action plus URL
    - To verify you have entered the correct Discovery parameters, execute the provided curl command. If it fails, re-check your parameter values.

4.4.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 and create it.
- Name the intent #Product_Information, 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.
- The dialog node should have a Return variable [1] set automatically to $webhook_result_1.
- This is the variable name you can use to access the result from the Discovery service query.
- You will also need to pass in the users question via the parameter input [2]. The key needs to be set to the value: ""
- If you fail to do this, Discovery will return results based on a blank query.

- Test in Assistant Tooling
    - From the Dialog panel, click the Try it button located at the top right side of the panel. Enter some 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.
    - So because we specified that $webhook_result_1.passages be the response, that value is displayed also. The response from the Discovery query will be stored in the $webhook_result_1 variable.

4.5.Create flow and configure node:

Integration of Watson assistant in Node-RED:

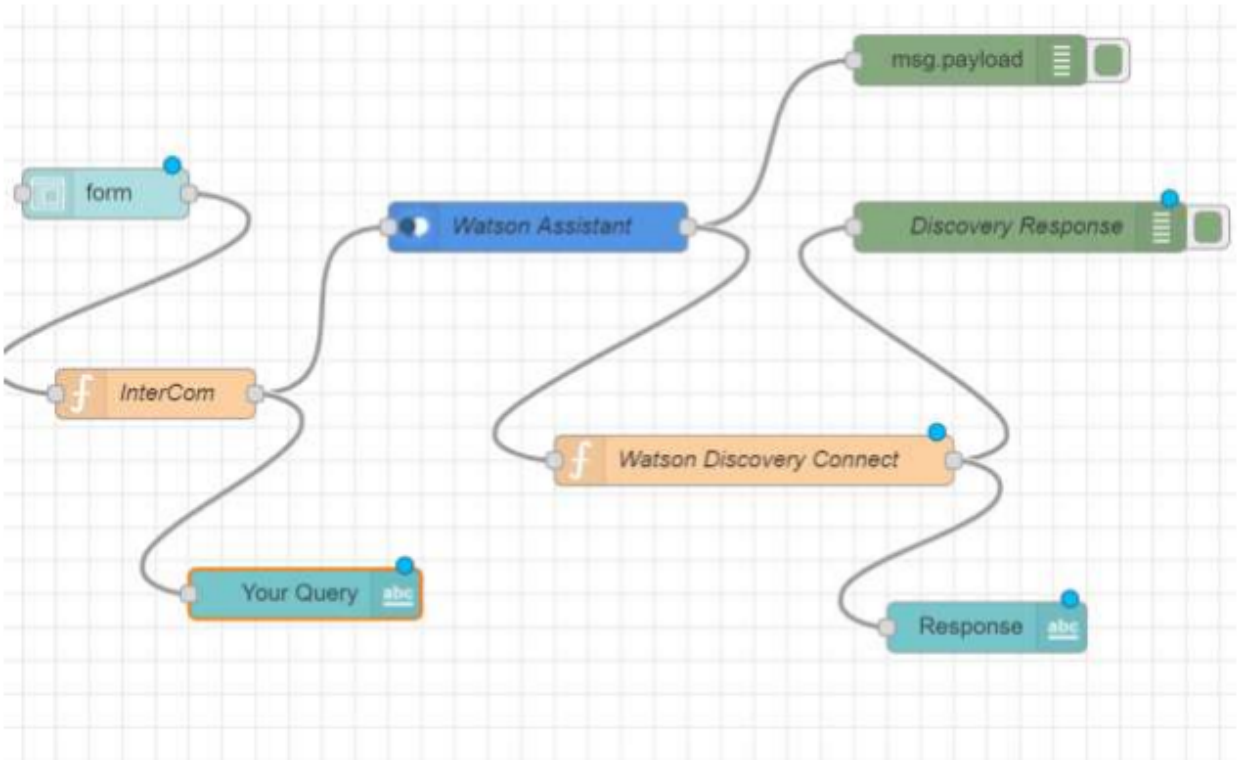
- Drag it and double-click on the Watson assistant node
- Give a name to your node and enter the username, password and workspace id of your Watson assistant service After entering all the information click on Done
- Drag template node on to the flow from the Input section. Drag Debug on to the flow from the output section
- For creating a web application UI we need “dashboard“ nodes which should be installed manually.
- Go to navigation pane and click on manage palette
- Click on install and Search for “node-red-dashboard” and click on install and again click on install on the prompt
- 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 group
- Give the label as “Enter your query”, 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” and "template"node from the “dashboard” section
- Drag these two nodes on to the flow
- Double click on the text node, change the label as “Your query” and click on Done.
- Connect the output of “input parsing” function node to text node and output of “Parsing” function node to the template node.
- Click on Deploy

5. FLOWCHART

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

- Template
- Assistant
- Debug
- Function
- Ui_Form and text



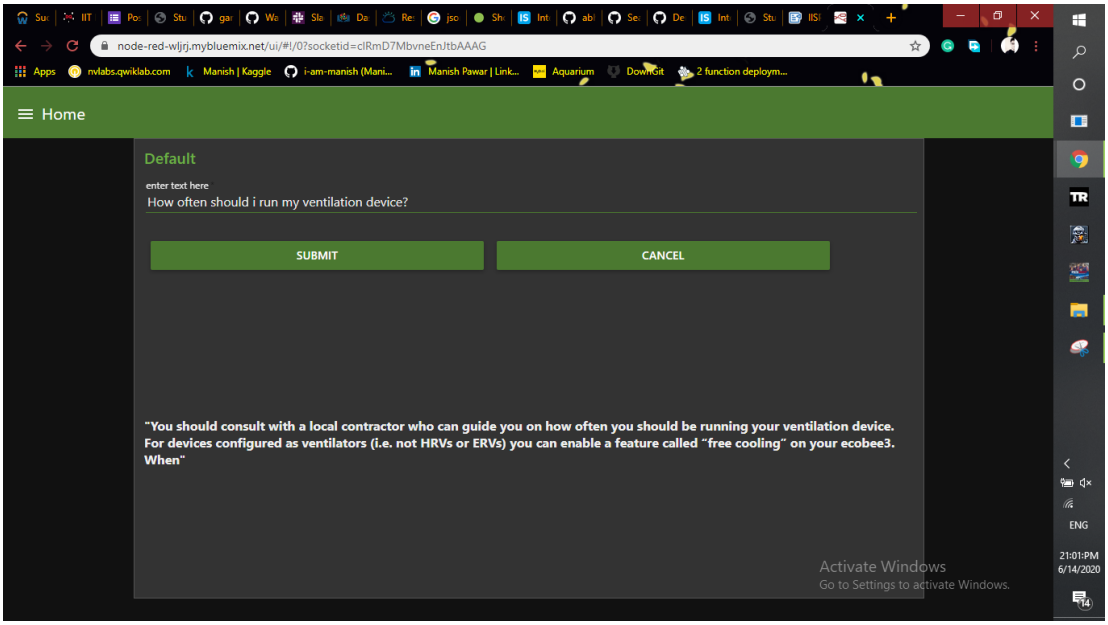
6. RESULTS

Finally our Node-RED dash board integrates all the components and displayed in the Dashboard UI by typing

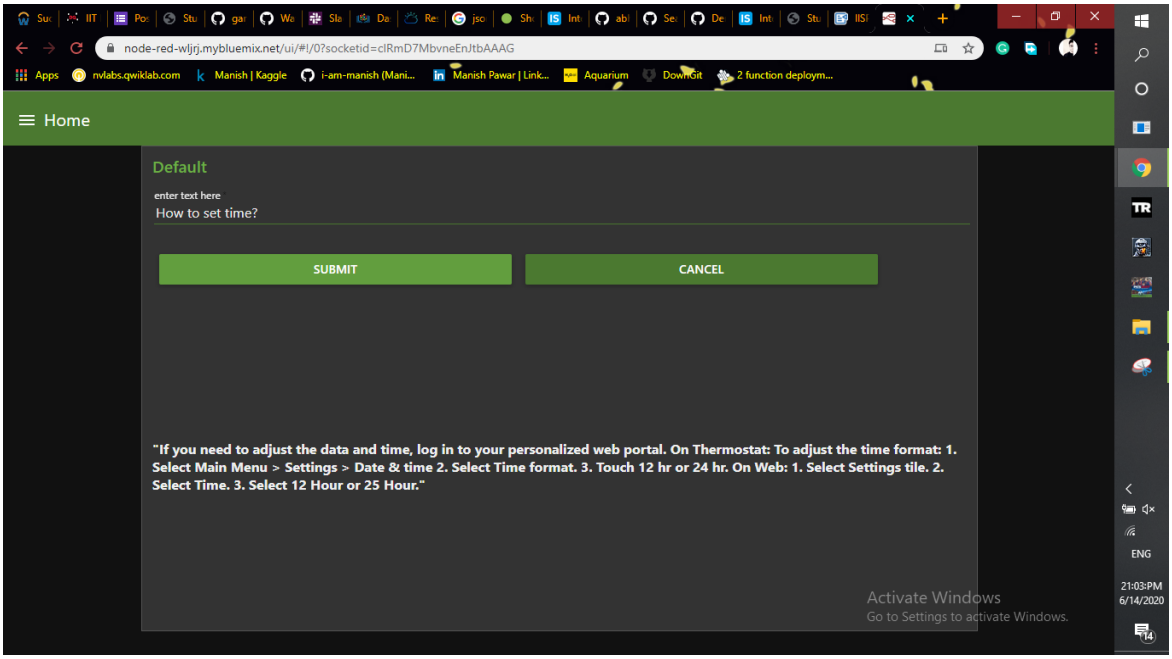
- https://node-red-wljrj.mybluemix.net/ui/ in browser.

Here are some pictures showing the responses of our queries by out bot.

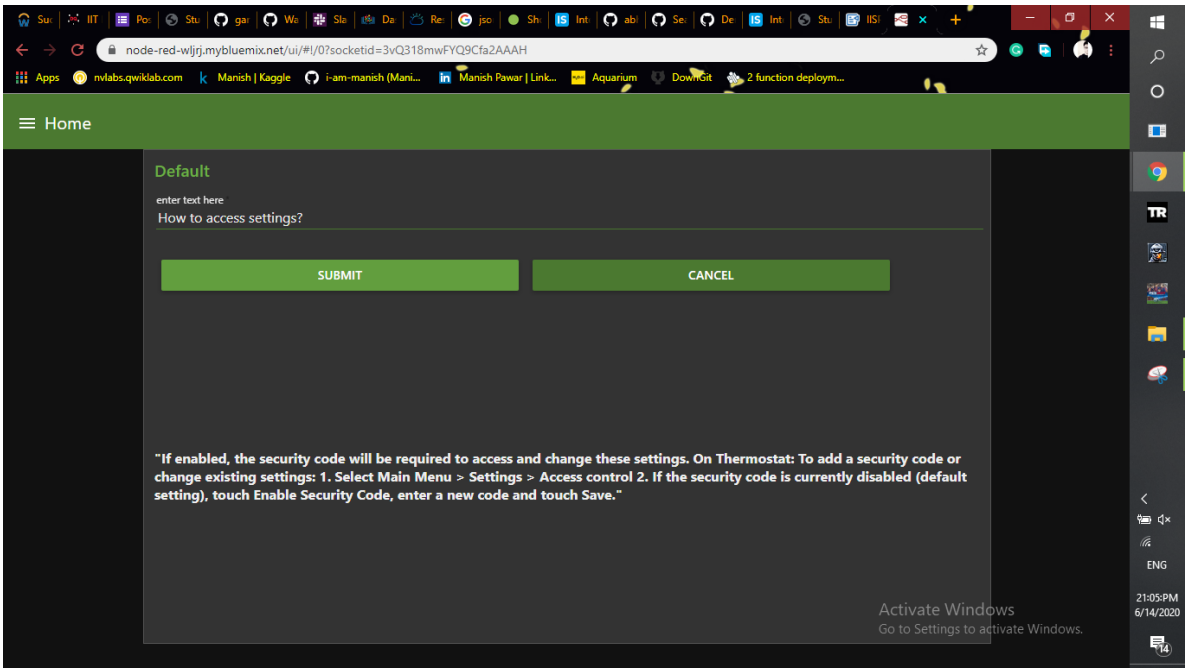
Query 1:



Query 2:



Query 3:



7. ADVANTAGES & DISADVANTAGES

Advantages:

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

Disadvantages:

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

8. APPLICATIONS

It can be deployed on many popular social media applications like facebook, slack, telegram.
Chatbot can deploy any website to clarify basic doubts of viewers.

9. CONCLUSION

By doing the above procedure and we have successfully created the Intelligent help desk smart chat bot using Watson assistant, Watson discovery, Node-RED and cloud-functions.

10. FUTURE SCOPE

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

11. BIBILIOGRAPHY

Source Code: Cloud Functions Action

```
1 * @param {object} params
2 * @param {string} params.iam_apikey
3 * @param {string} params.url
4 * @param {string} params.username
5 * @param {string} params.password
6 * @param {string} params.environment_id
```

```
7 * @param {string} params.collection_id
8 * @param {string}
  params.configuration_id
9 * @param {string} params.input
10 * @return {object}
11 const assert = require('assert'); const
  DiscoveryV1 =
12 require('watson-developer-cloud/discove
  ry/v1');
13 * @param Cloud Functions actions accept
  a single parameter, which
14 * @return The output of this action,
  which must be a JSON object.
15 function main(params) {
16 return new Promise(function (resolve,
  reject)
17 { let
18 discovery; if (params.iam_apikey){
  discovery =
19 new DiscoveryV1({
20 'iam_apikey': params.iam_apikey,
21 'url': params.url,
22 'version': '2019-03-25'
23 });
```

```
24 }
25 else {
26 discovery = new DiscoveryV1({
  'username':
27 params.username,
28 'password': params.password,
29 'url': params.url,
30 'version': '2019-03-25'
31 });
32 }
33 discovery.query({
34 'environment_id':
  params.environment_id,
35 'collection_id': params.collection_id,
36 'natural_language_query': params.input,
37 'passages': true,
38 'count': 3,
39 'passages_count': 3 },
40 function(err, data) {
41 if (err) {
42 return reject(err);
43 }
44 return resolve(data);
45 }); });
```

REFERENCES

- https://www.ibm.com/cloud/architecture/tutorials/cognitive_di scovery
- https://cloud.ibm.com/docs/assistant?topic=assistant-getting-s tarted
- https://developer.ibm.com/recipes/tutorials/how-to-create-a-wa tson-chatbot-on-nodered/
- http://www.iotgyan.com/learning-resource/integration-of-watso nassistant-to-node-red
- https://github.com/IBM/watson-discovery-sdu-with-assistant llSPS_INT_520_Intelligent Customer Help Desk with Smart Document Understanding