

PROJECT REPORT

Name: Shubham Dogra

E-mail : shubhamdogra076@gmail.com

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

**Category: Artificial Intelligence Internship at
SMARTINTERNZ**

Application ID: SPS_APL_20200002468

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 ANALYSIS

3.1 Block diagram

3.2 Hardware / Software designing

4 EXPERIMENTAL INVESTIGATIONS

5 FLOWCHART

6 RESULT

7 ADVANTAGES & DISADVANTAGES

8 APPLICATIONS

9 CONCLUSION

10 FUTURE SCOPE `

11 BIBILOGRAPHY

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 Requirements

IBM Cloud, IBM Watson, Node-Red, NodeJS

➤ Functional Requirements

IBM Cloud

➤ Technical Requirements

Artificial Intelligence, Machine Learning, Watson AI, NodeJS

➤ Software Requirements

Watson Assistant, Watson Discovery, Node-Red

➤ Project Deliverables

Intelligent Chatbot with Smart document understand

➤ Project Team

Shubham Dogra

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 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.

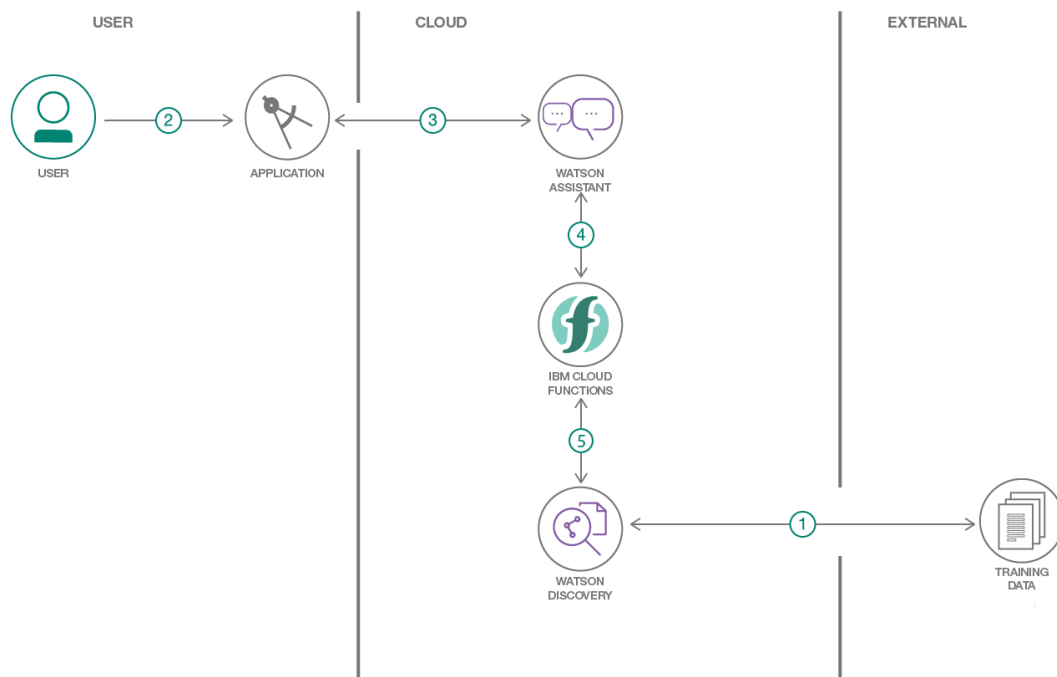
2.2 Proposed solution:

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. THEORITICAL ANALYSIS

3.1 Block/Flow Diagram

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



3.2 Hardware / Software designing:

1. Create IBM Cloud services
2. Configure Watson Discovery
3. Create IBM Cloud Functions action
4. Configure Watson Assistant
5. Create flow and configure node
6. Deploy and run Node Red app.

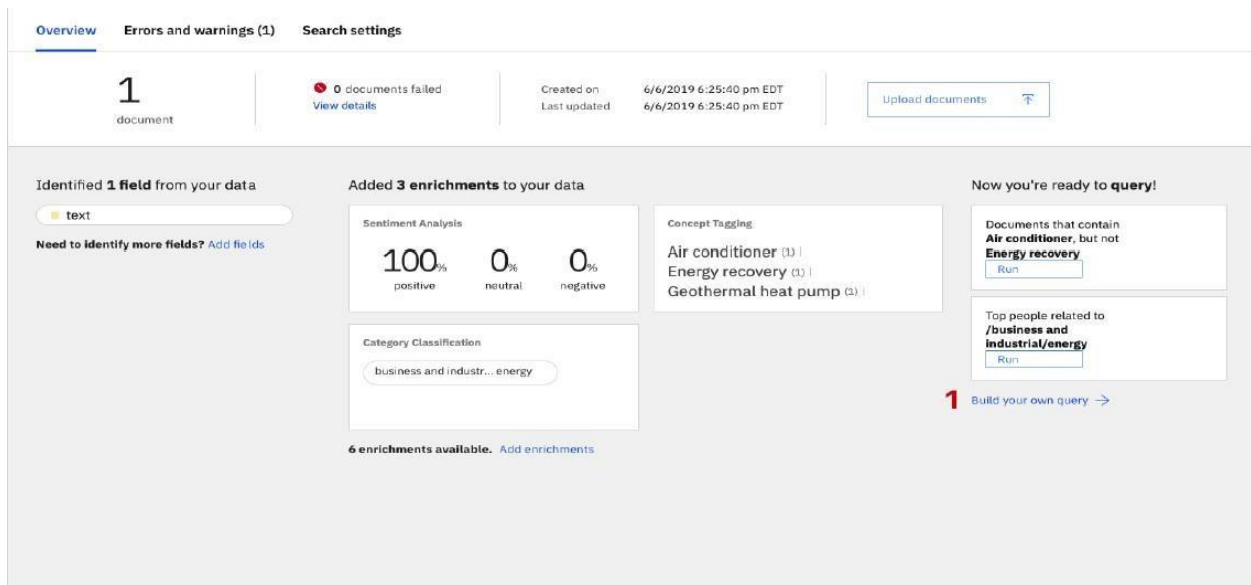
4. EXPERIMENTAL INVESTIGATIONS

1. Create IBM Cloud services Create the following services:

- Watson Discovery
- Watson Assistant
- Node Red

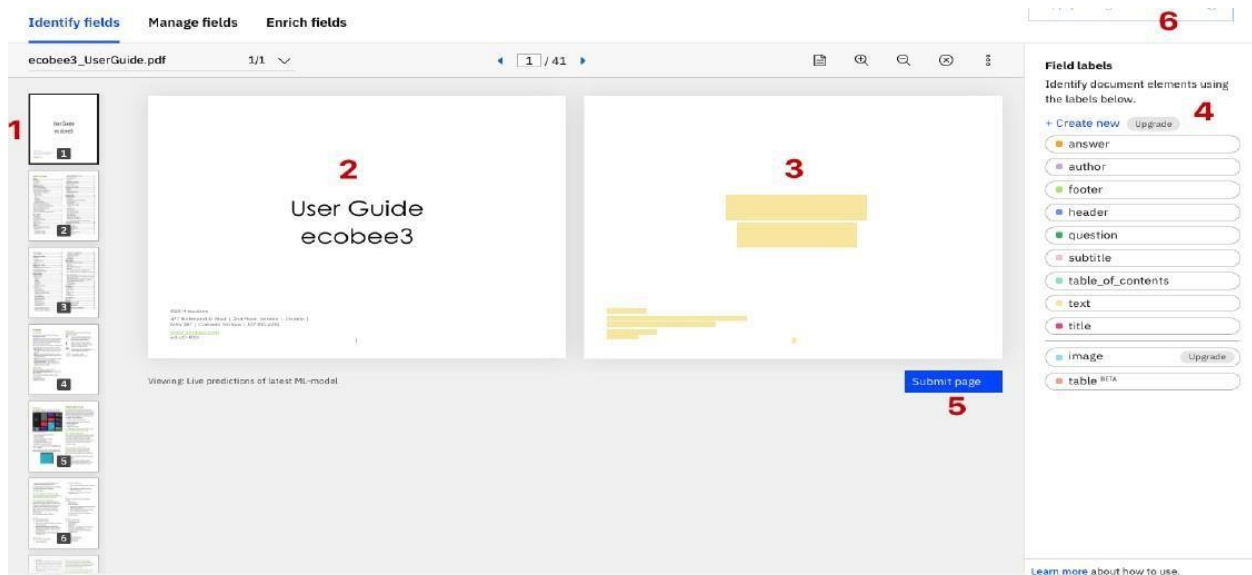
2. Configure Watson Discovery

- 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 repository. ➤ The Ecobee is a popular residential thermostat that has a wifi interface and multiple configuration options.
- Before applying SDU to our document, let's do some simple queries on the data so that we can compare it to results found after applying SDU.
- Enter queries related to the operation of the thermostat and view the results.
- Annotate with SDU. Now let's apply SDU to our document to see if we can generate some better query responses. From the Discovery collection panel, click the Configure data button (located in the top right corner) to start the SDU process. ➤ Here is the layout of the Identify fields tab of the SDU annotation panel:



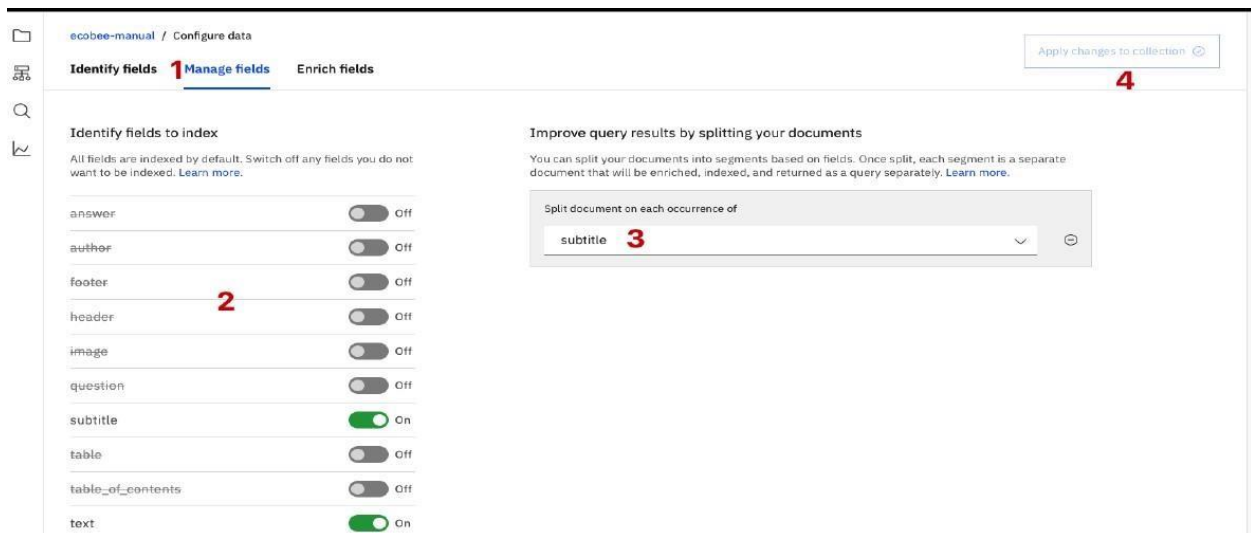
➤ The goal is to annotate all of the pages in the document so Discovery can learn what text is important, and what text can be ignored. Follow the following instructions to achieve this task:

- [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.



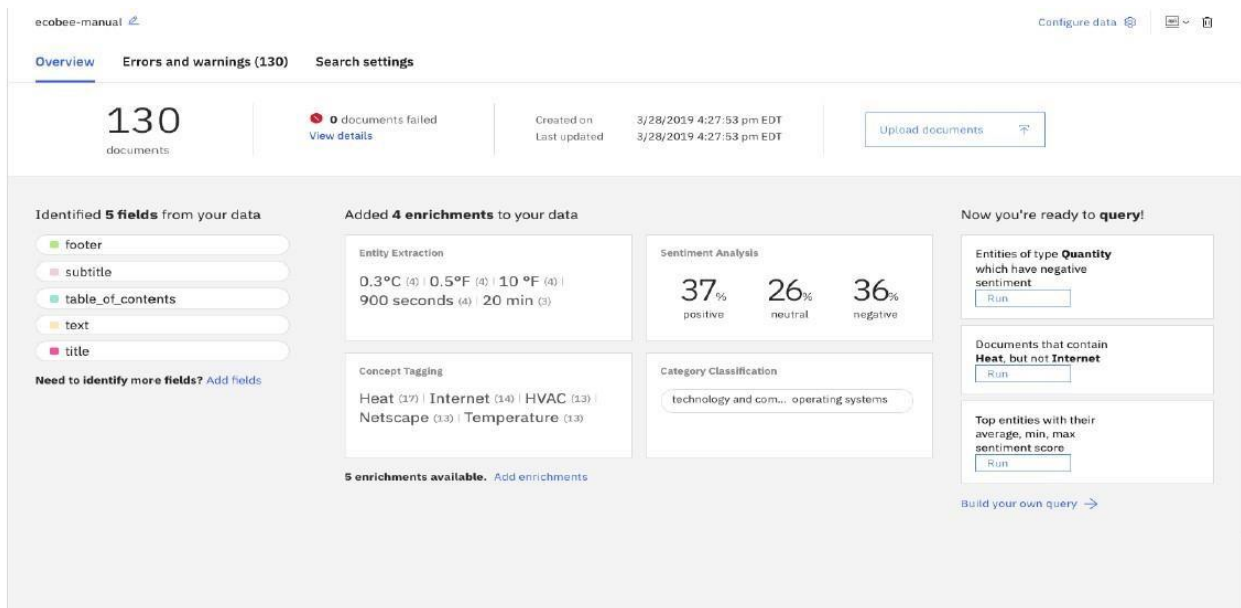
- As you go through the annotations one page at a time, Discovery is learning and should start automatically updating the upcoming pages. Once you get to a page that is already correctly annotated, you can stop, or simply click Submit [5] to acknowledge it is correct. The more pages you annotate, the 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 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.



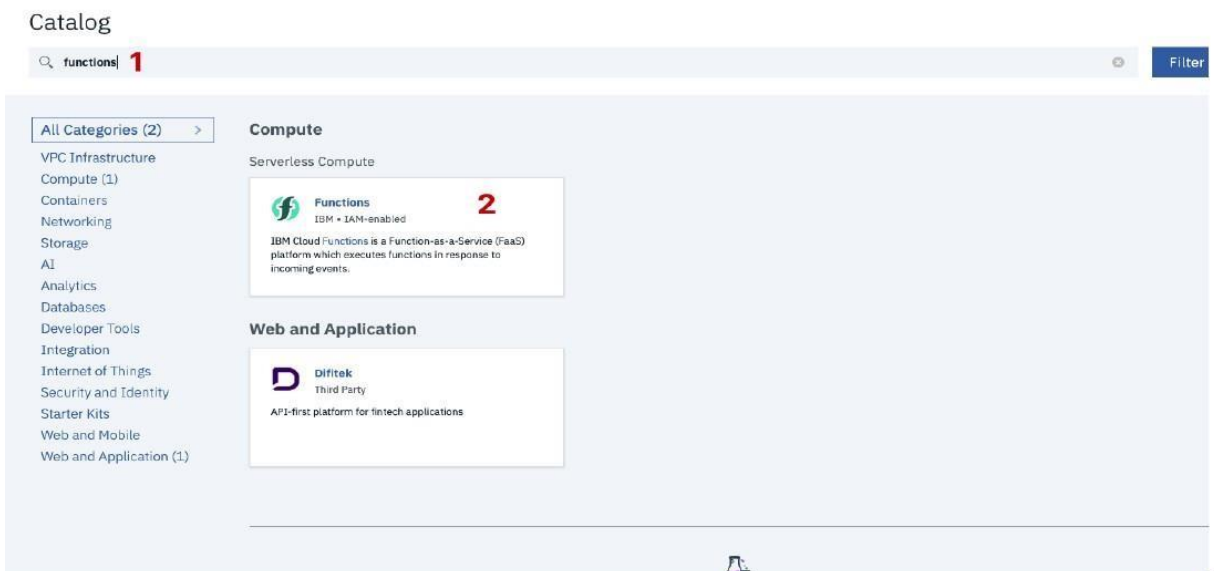
- [2] Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except subtitles and text.
- [3] is telling Discovery to split the document apart, based on subtitle.
- Click [4] to submit your changes.
- Once again, you will be asked to reload the document.
- 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.

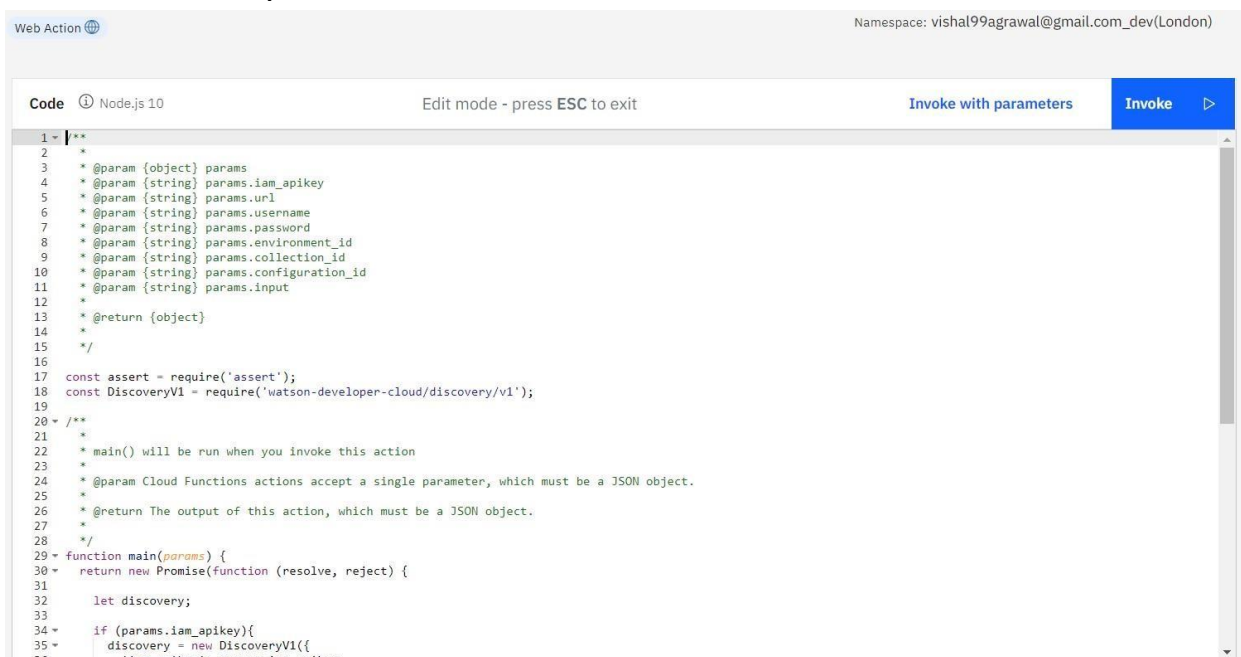


3.Create IBM Cloud Functions action

- 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.



Web Action 🌐 Namespace: vishal99agrawal@gmail.com_dev(London)

Code 🕒 Node.js 10 Edit mode - press ESC to exit Invoke with parameters Invoke ▶

```

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  function main(params) {
30    return new Promise(function (resolve, reject) {
31      let discovery;
32
33      if (params.iam_apikey){
34        discovery = new DiscoveryV1({
35          'iam_apikey': params.iam_apikey,

```

- If you press the Invoke button [3], it will fail due to credentials not being defined yet. We will do this next.
 - Select the Parameters tab:
- Add the following keys:

- url
- environment_id
- collection_id
- iam_apikey

The screenshot shows the 'Parameters' tab for a function named 'discovery-function'. The parameters are listed in a table with two columns: 'Parameter Name' and 'Parameter Value'.

Parameter Name	Parameter Value
url	"https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/abbf4655-b6c3-4a8f"
environment_id	"61224590-175c-41c8-ad3a-8a085fa1ecce"
collection_id	"d3777ccb-b5eb-4a2f-af1b-aea94707c2e8"
iam_apikey	"ssn2kRrGPCnirZpwFnPonTau8AJrOYtWYwCJ9cFg-gsi"

- 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 [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.

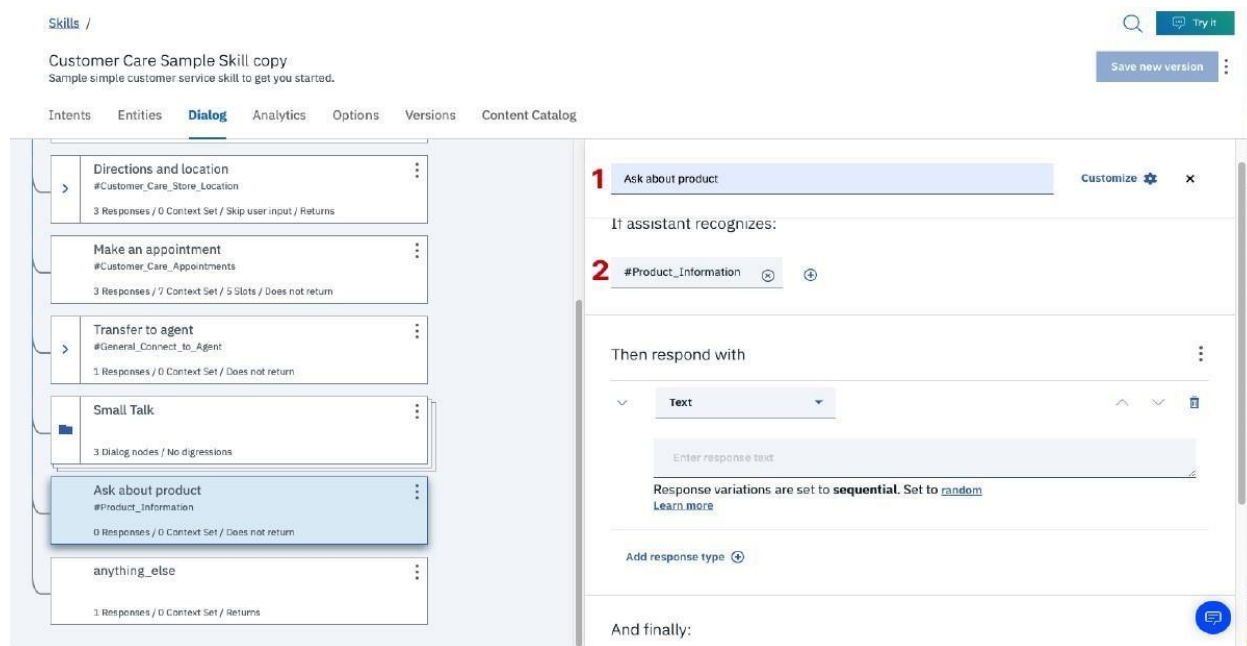
The screenshot shows the Watson Assistant 'Intents' tab for a skill named '#Product_Information'. The interface includes a header with a back arrow, the skill name, and a 'Try it' button. Below the header, there are fields for 'Intent name' (pre-filled with '#Product_Information'), 'Description (optional)' (pre-filled with 'User wants help using the thermostat'), and 'Add user example' (with a text input field). At the bottom, there is a table of 'User examples (3)' with columns for the example text, 'Added' time, and 'Conflicts' status. The examples are 'How do I access the settings', 'How do I set the time', and 'How do I turn on the heater', all added 2 hours ago with 0 conflicts.

Intent name	Description (optional)	Add user example
#Product_Information	User wants help using the thermostat	[type a user example here]

User examples (3)	Added	Conflicts
How do I access the settings	2 hours ago	0 conflicts
How do I set the time	2 hours ago	0 conflicts
How do I turn on the heater	2 hours ago	0 conflicts

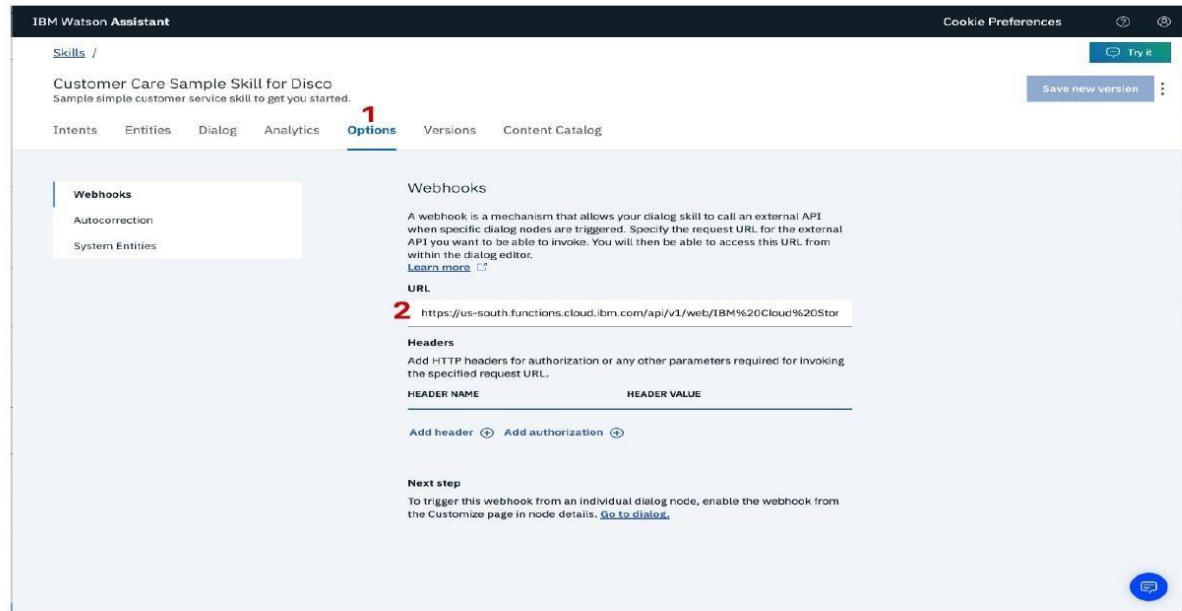
- Click the Create intent button.
- Name the intent #Product_Information, and at a minimum, enter the following example questions to be associated with it.
- Create new dialog node

- 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
- Set up access to our WebHook for the IBM Cloud Functions action you created in Step #4(Create IBM Clouds Function).



- 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 will also need to pass in the users question via the parameter input [2]. The key needs to be set to the value: "<?input.text?>"
- If you fail to do this, Discovery will return results based on a blank query.

-

Optionally, you can add these responses to aid in debugging:

Return variable

`$webhook_result_1`

Then respond with

	IF ASSISTANT RECOGNIZES	RESPOND WITH		
1	<code>\$webhook_result_1</code>	<code>\$webhook_result_1</code>		
2	<code>anything_else</code>	Try again later		

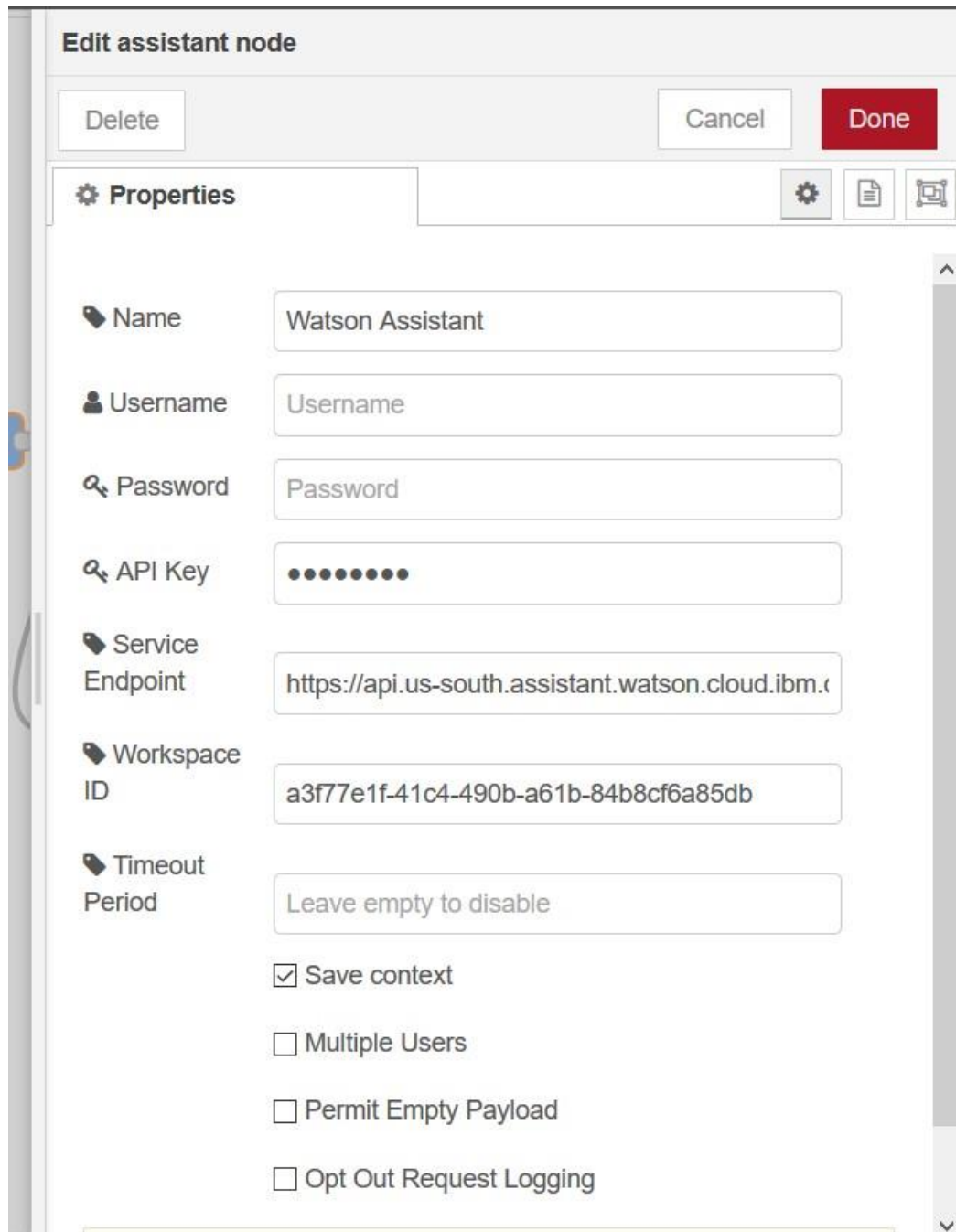
[Add response](#) 

➤ 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.
- So 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.

5. Create flow and configure node:

► Integration of Watson assistant in Node-RED:



Edit assistant node

Delete Cancel Done

Properties

Name Watson Assistant

Username Username

Password Password

API Key

Service Endpoint https://api.us-south.assistant.watson.cloud.ibm.com

Workspace ID a3f77e1f-41c4-490b-a61b-84b8cf6a85db

Timeout Period Leave empty to disable

☒ Save context

☐ Multiple Users

☐ Permit Empty Payload

☐ Opt Out Request Logging

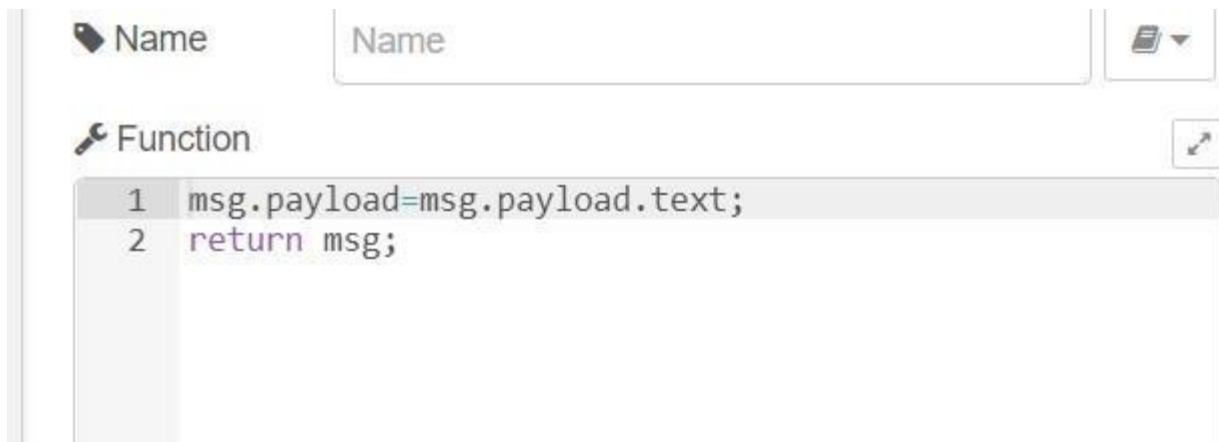
- Double-click on the Watson assistant node

-

- 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
- 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
- Double 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



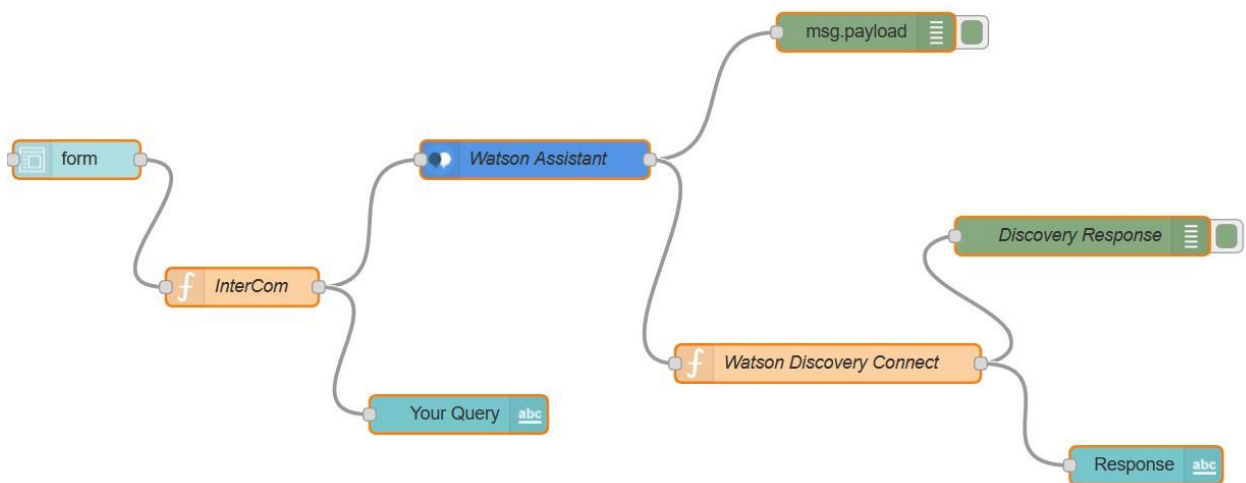
- Click on done
- Connect the form output to the input of the function node and output of the function to input of assistant node
- Search for “text” and "template" node from the “dashboard” section
- Drag these two nodes on to the flow
- Double click on the text node, change the label as “Your query” 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 pallette and install dashboard.
- Now, Create the flow with the help of following node:
 - Template
 - Assistant
 - Debug

-
- Function
- Ui_Form
 - Ui_Text

Flowchart of node-red app will look like this:



6. RESULTS

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

<https://node-red-lhmea.mybluemix.net/ui/#!/0?socketid=DnFdXvVIIBhE8PvUAAAG> in browser.

➤ Here are some pictures showing the responses of our queries by the chat bot : ● Query 1

Smart LapBot

Enter the Query:: *

how are you?

SUBMIT

CANCEL

Your Query

how are you?

Response

Hi there. It's a beautiful afternoon



- Query 2

●

The screenshot shows a chatbot interface for 'Smart LapBot'. At the top, the text 'Smart LapBot' is displayed in blue. Below it, a text input field contains the query 'How Do i configure WI-Fi?'. To the right of the input field is a vertical scrollbar. Below the input field are two blue buttons: 'SUBMIT' and 'CANCEL'. Underneath the buttons, the text 'Your Query' is followed by the query 'How Do i configure WI-Fi?'. Further down, the text 'Response' is followed by a detailed answer: 'Your ecobee3 supports Wi-Fi 802.11 b/g/n. Wi-Fi is normally configured during initial setup. You may, however, be required to reconfigure the settings if your Wi-Fi network settings change. On Thermostat: Select Main Menu > Settings > Wi-Fi.'

●

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

```
/**
```

```
*
```

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

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-chatbot-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>
IISPS_INT_520_Intelligent Customer Help Desk with Smart Document Understanding