Summer Internship Project Report

Internship Title: Intelligent Customer Help Desk With Smart Document Understanding – SB51613
Project Title: Intelligent Customer Help Desk With Smart Document Understanding
Project Duration: 06/06/2020 - 06/07/2020
Category: Artificial Intelligence

Submitted by: Shubham
Intern ID: IISPS-INT-2712

Email: **shubhambit92@gmail.com**

From:

SmartBridge Educational Services Pvt Ltd.
Plot No 132, Above DCB bank, 2nd floor,
Bapuji Nagar, Habsiguda,
Nacharam Main Road, Hyderabad – 500076

Date: 21/05/2020.

Dear **Shubham**

SmartBridge Educational Services Pvt Ltd, is pleased to offer a training cum internship opportunity. During this period you would be associated with our mentors and The Smart Practice School Platform.

For further details you can contact us on +91 8499004200.

Thanks and Regards,

Ch. Jaya Prakash
Program Manager – SIP2020,
Date: 16/05/2020.

Preface

This report documents is the work done during the summer internship at **SmartBridge Educational Services PVT Ltd**, for the Intelligent Customer Help Desk Chat Bot With Smart Document Understanding under the supervision of **mentors of SmartBridge and smart practice school platform**. The report will give an overview of the tasks completed during the period of internship with technical details. Then the results obtained are discussed and analyzed. I have tried my best to keep report simple yet technically correct. I hope I succeed in my attempt.

Shubham

Index

1. INTRODUCTION

1.1. Overview

The typical customer care chatbot 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 predetermined 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 preloaded with the device's owner's manual. So now, instead of "Would you like to speak to a customer representative?" we can return relevant sections of the owner's 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 owner's manual is important and what is not. This will improve the answers returned from the queries.

Project Requirements: Node.js, IBM Cloud, IBM Watson
Functional Requirements: IBM cloud

Technical Requirements: AI, NODE-RED APP, NODE.JS 10

Software Requirements: Watson assistant, Watson discovery.

Project Deliverables: Smartinternz Internship

Project Team: Shubham

Project Duration: 23.5 Days

1.2. Purpose

- 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 & deploy the same on IBM Cloud Platform.

2. LITERATURE SURVEY

2.1. Existing Problem:

The typical customer care chatbot 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.

2.2. Proposed Solution

Steps

1. Create IBM Cloud services
2. Configure Watson Discovery
3. Create IBM Cloud Functions action.
4. Create a Node red flow to connect all the services together.
5. Configure Watson Assistant.
6. Create flow and configure node
7. Deploy and run Node Red app.

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 repo.

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. 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.

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, then select the Functions card:

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, keep the default package and select the Node.js 10 runtime.

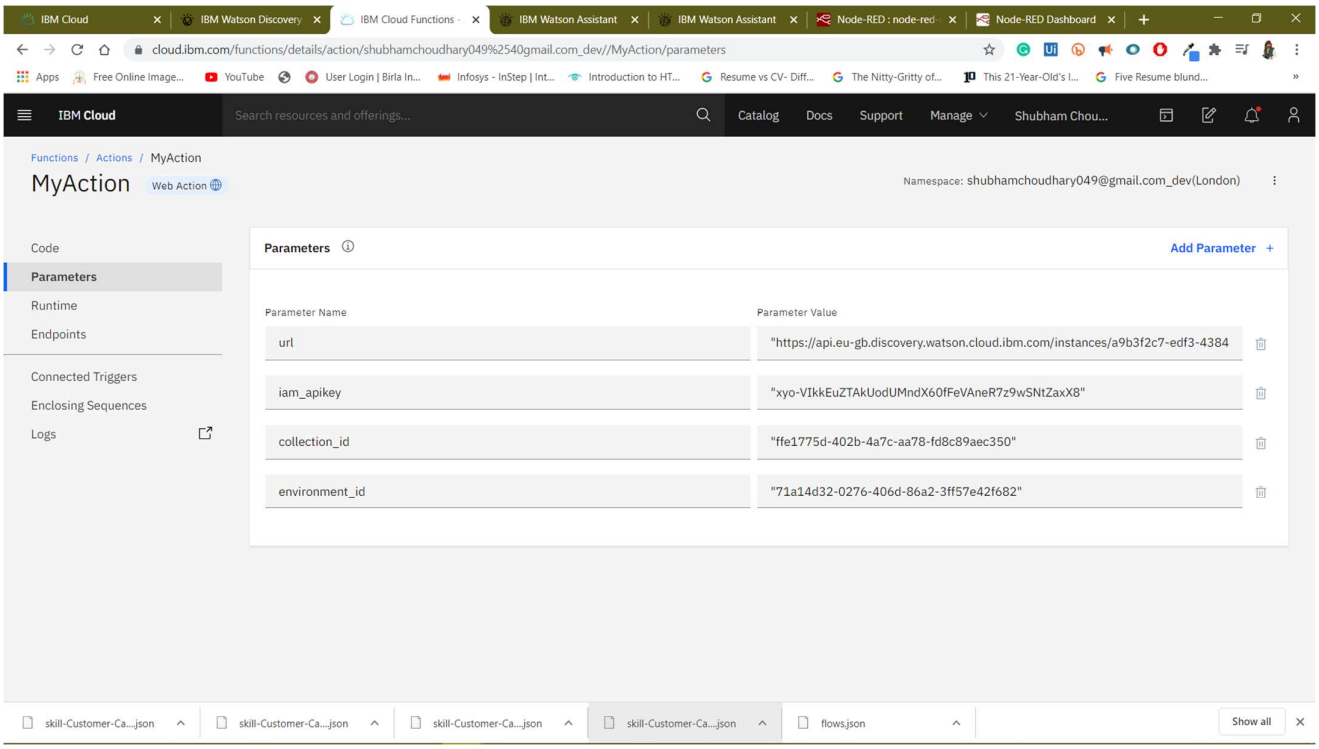
Click the Create button to create the action.
Once your action is created, click on the Code tab:

In the code editor window, cut and paste in the code from the disco-action.js file found in the action's 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.

If you press the Invoke button, it will fail due to credentials not being defined yet. We'll do this next. Select the Parameters tab:

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 Discovery service:
Next, go to the Endpoints panel:
Click the checkbox for Enable as Web Action. This will generate a public endpoint URL.
Take note of the URL value, 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. 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 Product. From the Customer Care Sample Skill panel, select the Intents tab.

Click the Create intent button. Name the intent #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 tab, then click on the drop-down menu for the Small Talk node, and select the Add node below option.

Name the node "Enquiry" and assign it our new intent. This means that if Watson Assistant recognizes a user input such as "How to configure my time zone?" 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:

Enter the public URL endpoint for your action. Return to the Dialog tab, and click on the Enquiry 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 set automatically to $webhook_result_1. This is the variable name you can use to access the result from the Discovery service query.

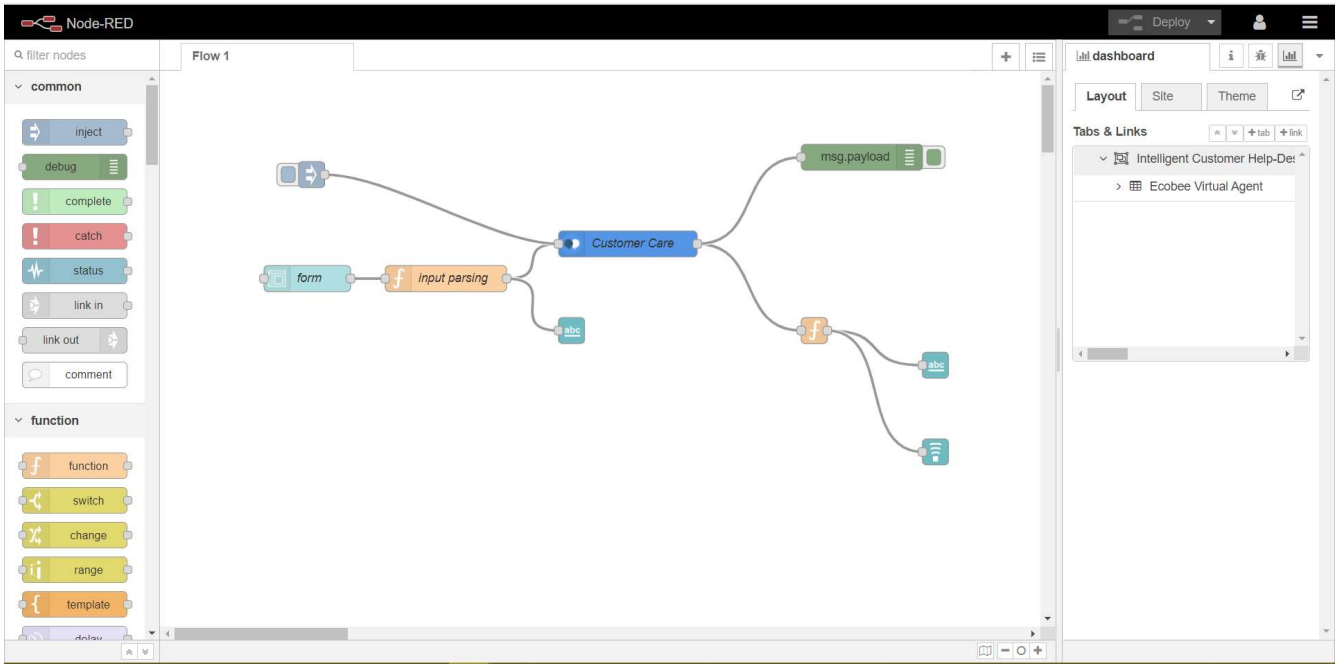
Test in Assistant Tooling

From the Dialog panel, click the Try it button located at the top right side of the panel. Enter some user input: Note that the input " How to adjust my schedule?" has triggered our Enquiry dialog node, which is indicated by the #Enquiry 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.

5. Create flow and configure node:

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

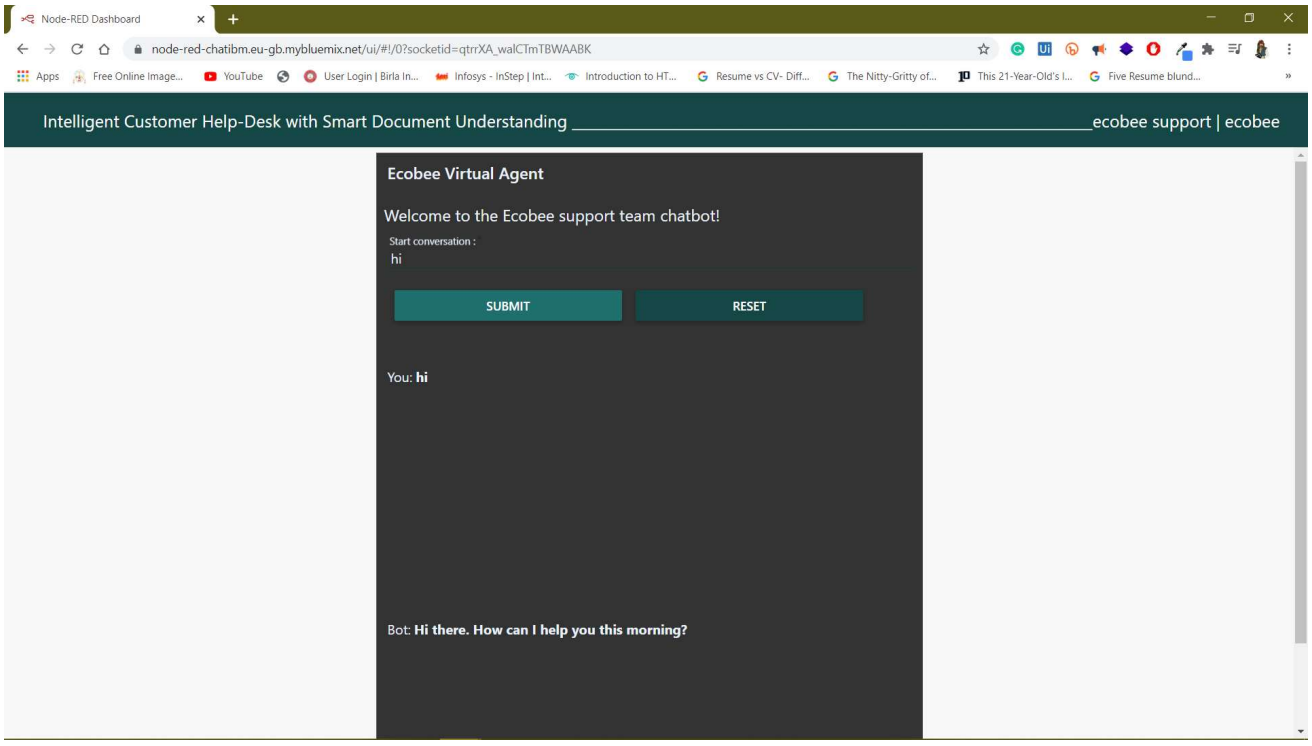
- Inject
- Assistant
- Debug
- Function
- UI_Form
- UI_Text
- Audio out



6. Deploy and run Node Red app.

Deploy the Node Red flow.
Then copy the link url upto .net/ and paste at anew tab by ui at the end of the url, like this,

https://node-red-chatibm.eu-gb.mybluemix.net/ui/

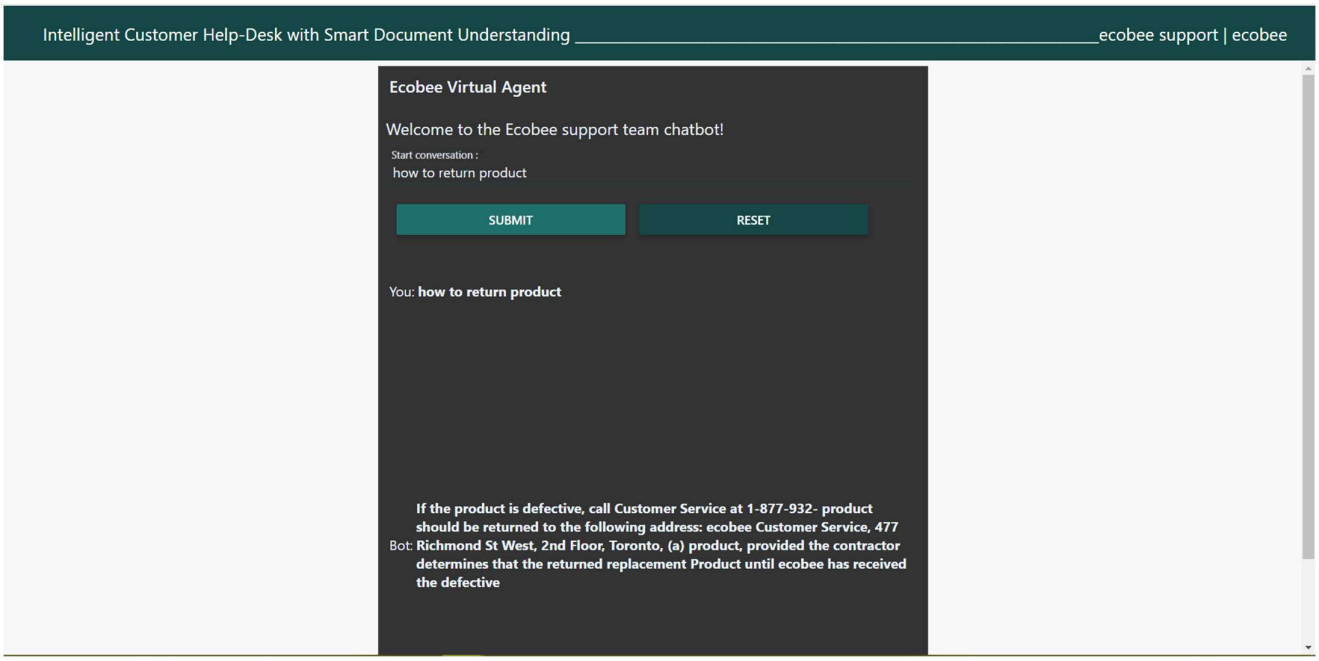


3. THEORITICAL ANALYSIS

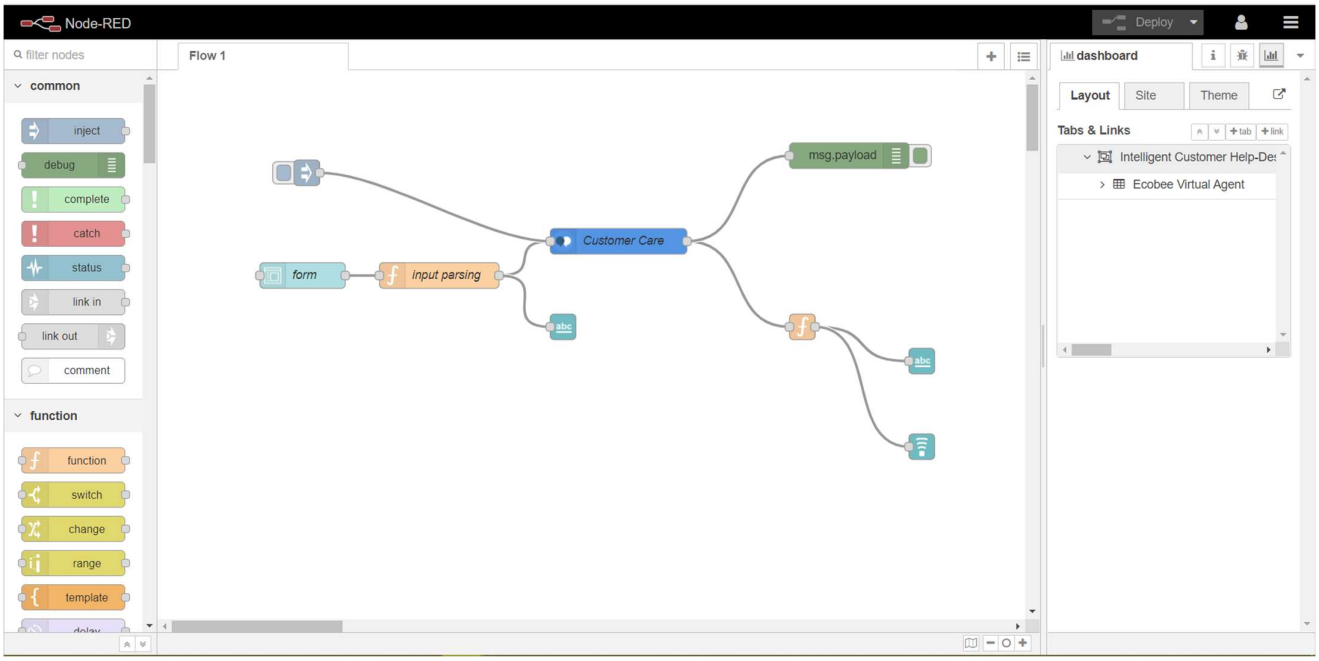
3.1 Hardware and Software Designing

Project Requirements: Node.js, IBM Cloud, IBM Watson
Functional Requirements: IBM cloud
Technical Requirements: AI, NODE-RED APP, NODE.JS 10
Software Requirements: Watson assistant, Watson discovery.

A node red flow is made for the chatbot. Which is given below.
https://node-red-chatibm.eu-gb.mybluemix.net/ui/#!/0?socketid=wnCozjZYa4KuqebRAACL



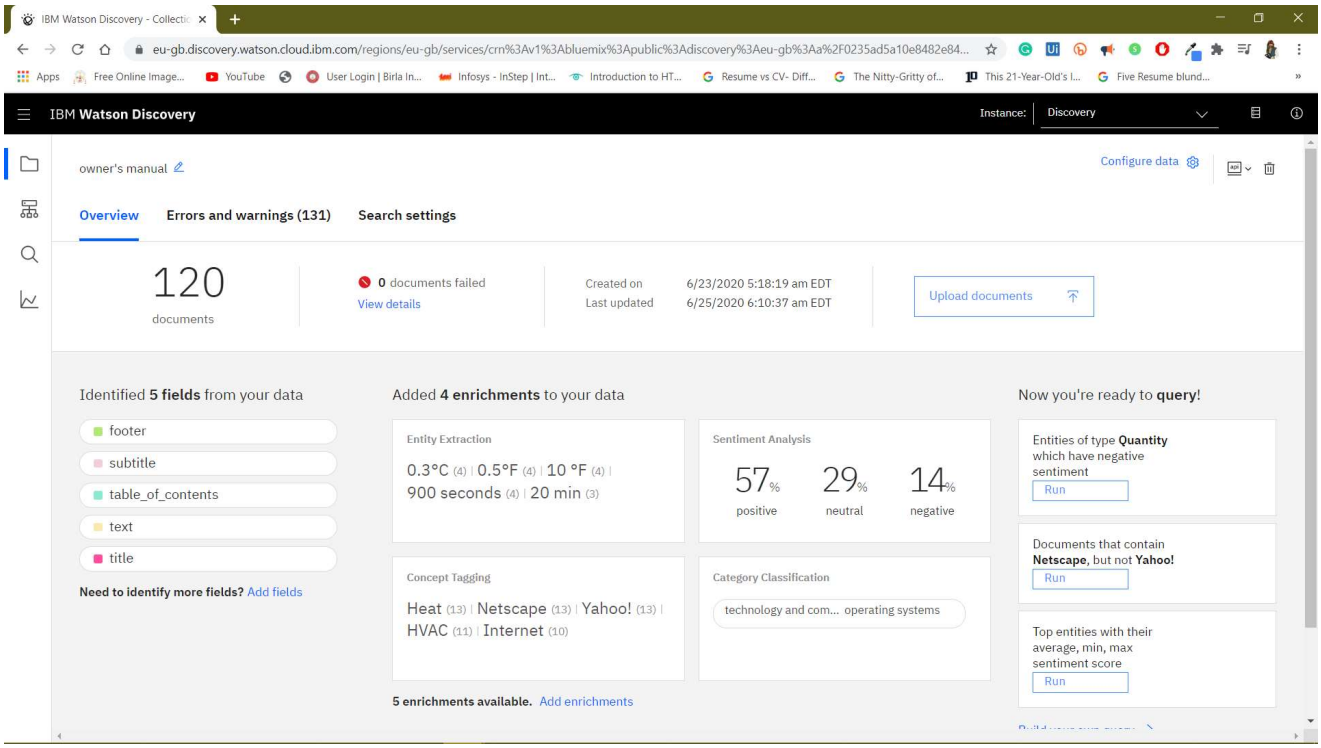
Demonstration of node red flow



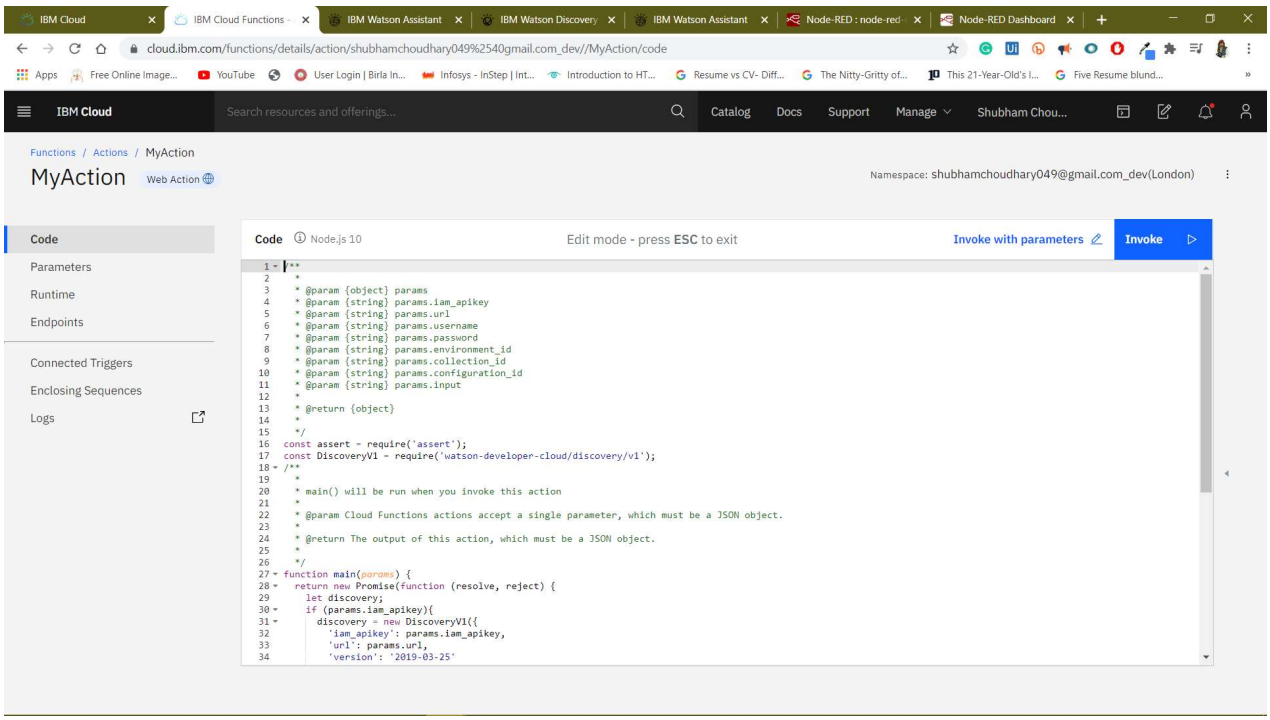
4. EXPERIMENTAL INVESTIGATIONS

- On IBM Watson cloud using Watson assistant integrated with Watson discovery build a chatbot to answer all the queries of a Ecobee users.
- Pre-load discovery with Ecobee Owner's Manual.
- Use Smart Document Understanding for better search results.

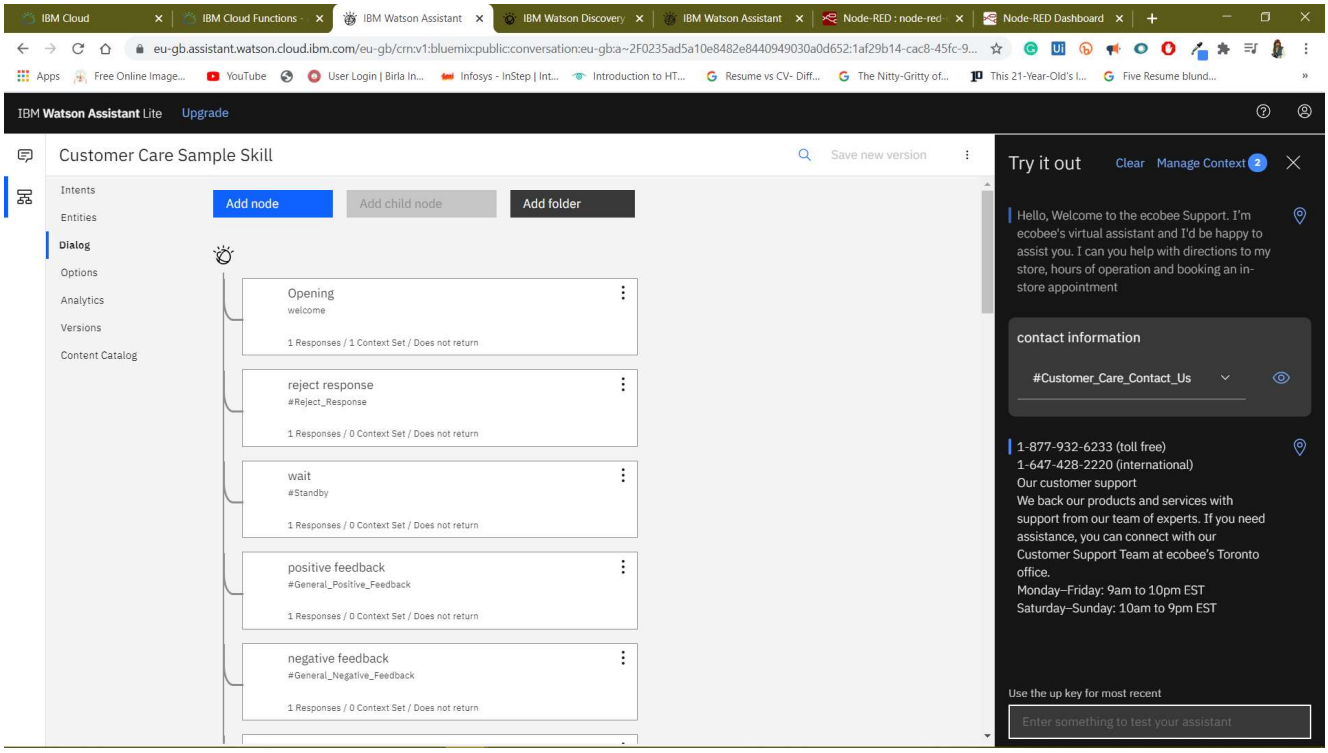
- Watson Discovery – to understand the document:



- Cloud Function – action generation:

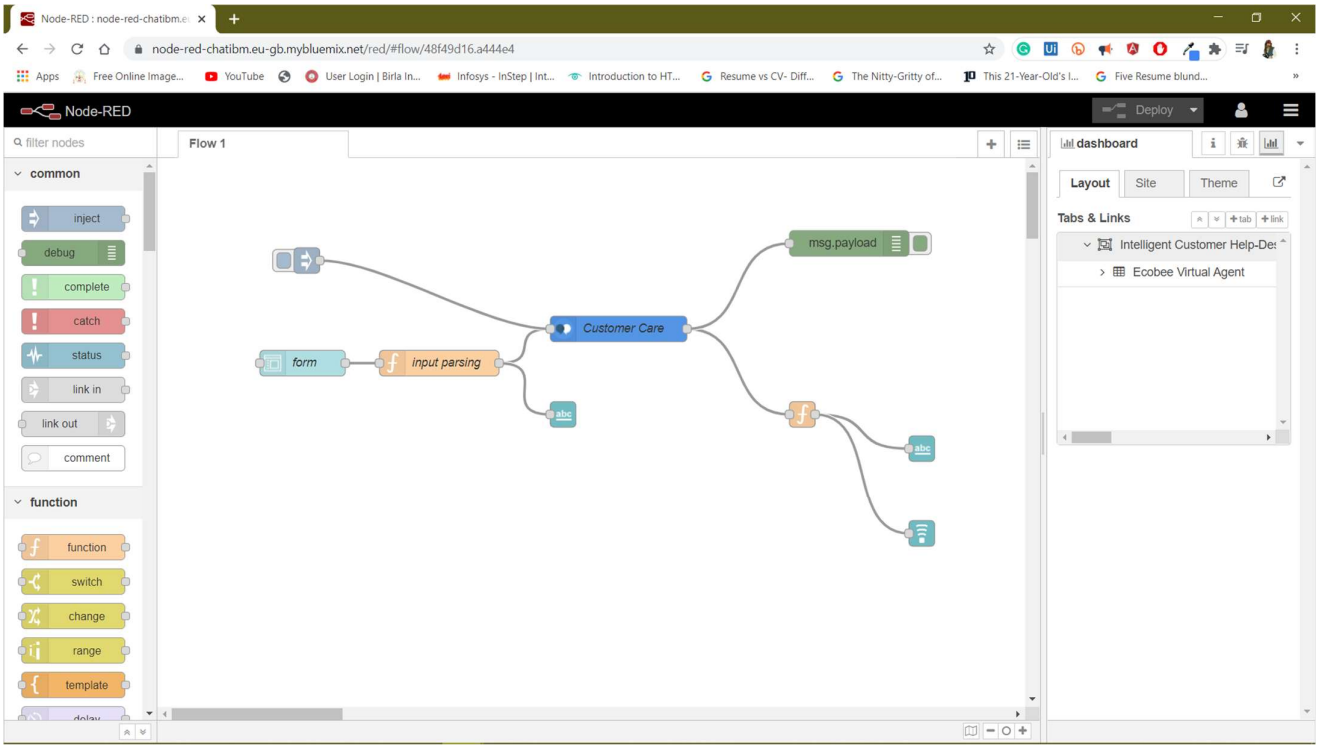


- Watson Assistant – to build chatbot:

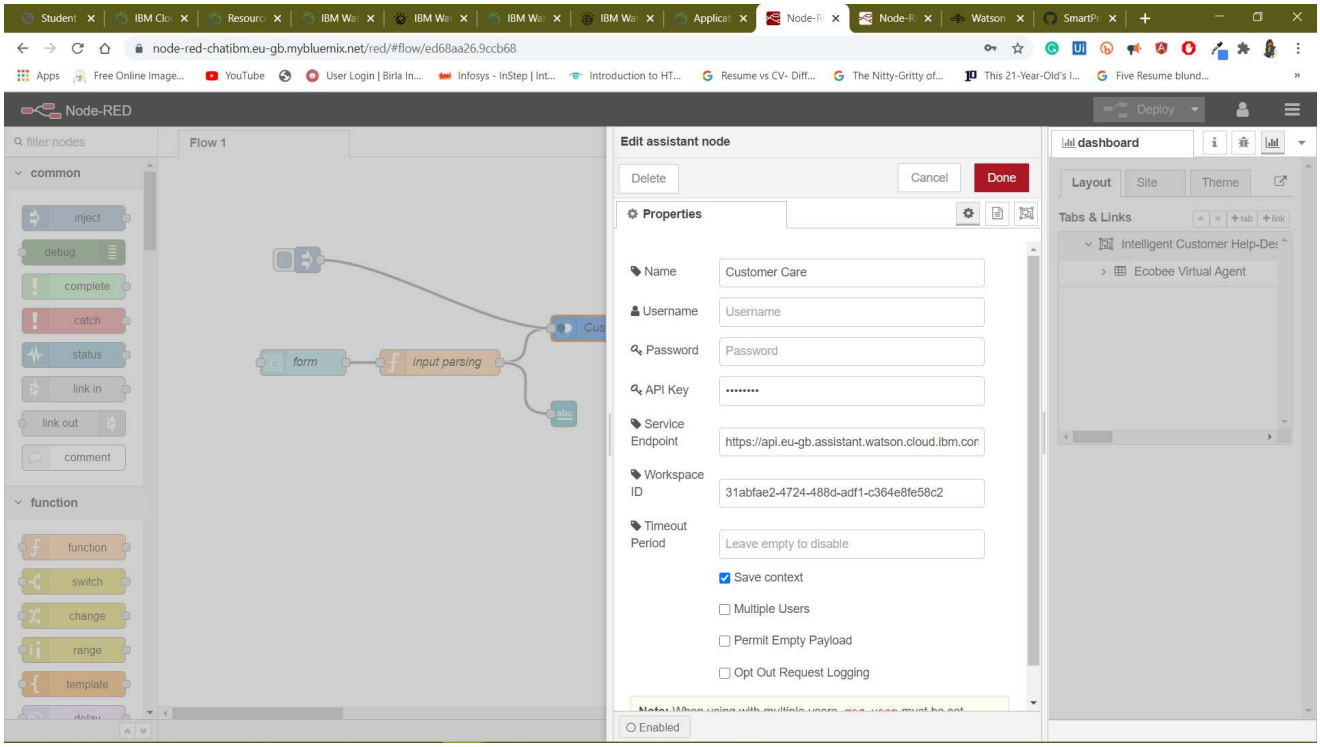


• Create cloud foundry app
https://node-red-chatibm.eu-gb.mybluemix.net/red/#flow/ed68aa26.9ccb68

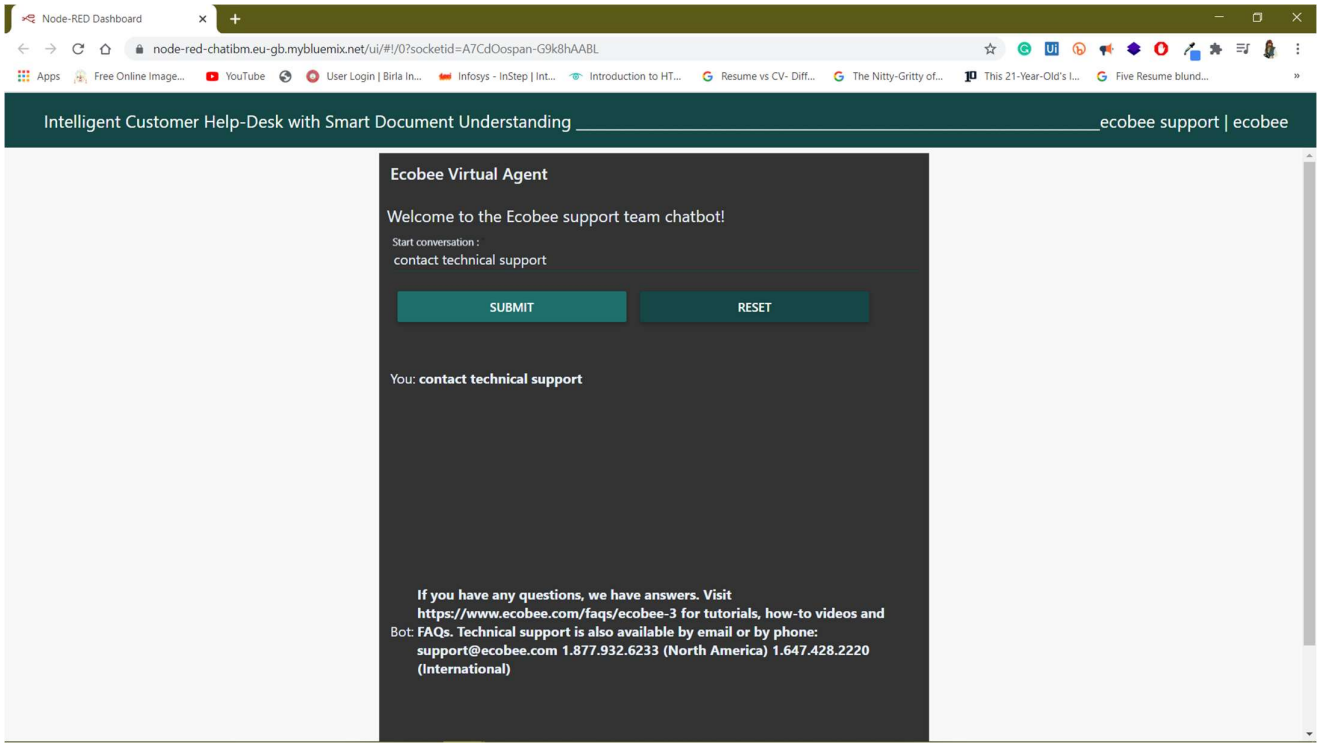
• Node-Red flow – UI creation (Flow based tool):



• Then add API key, workspace Id and url.

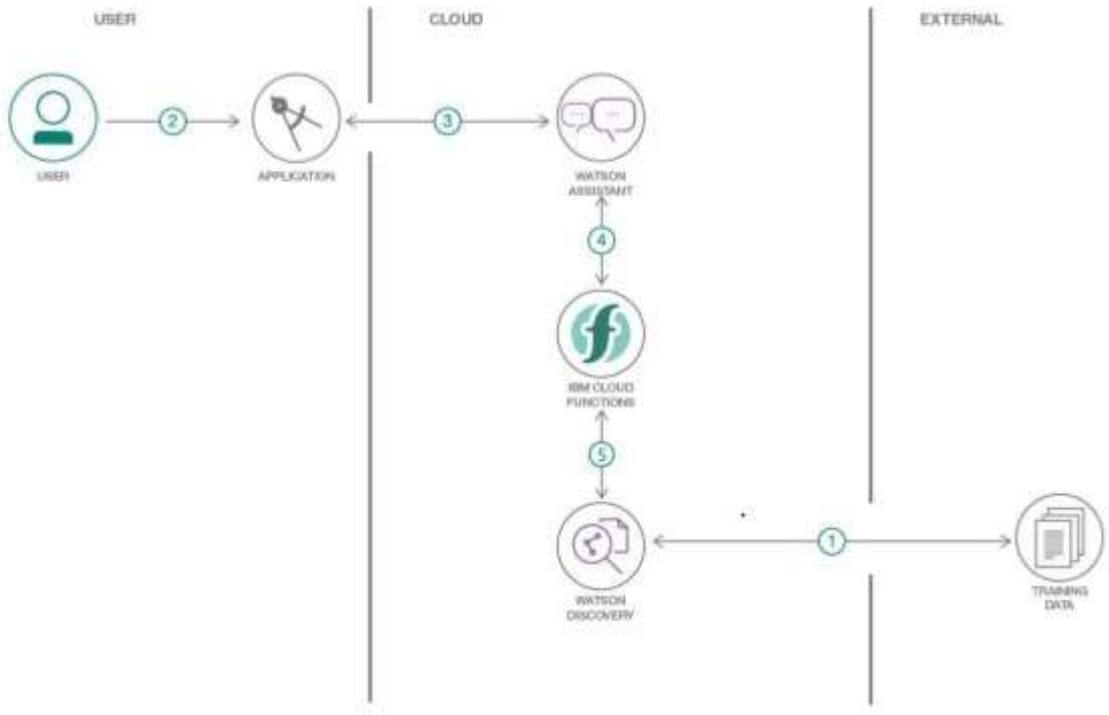


• After deploying the chatbot from dashboard UI can be seen.



5. FLOWCHART

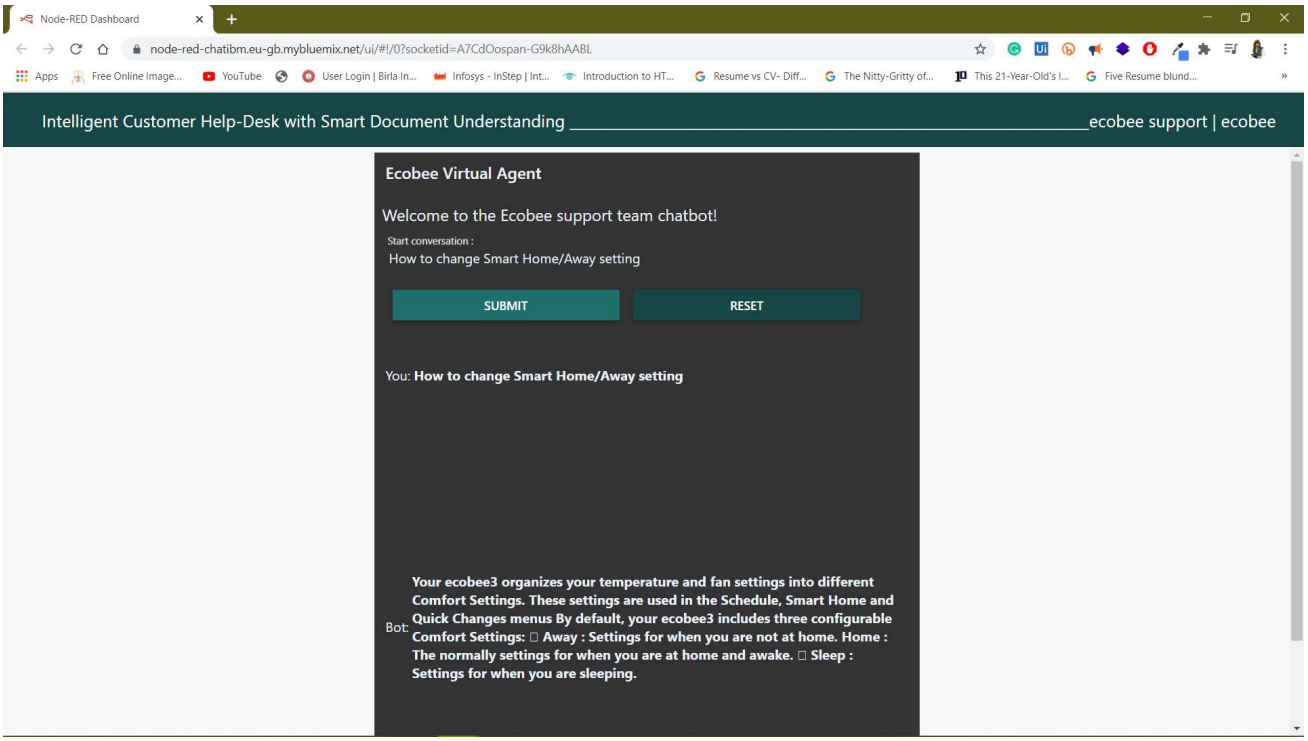
This is the flow how we are going to do the tasks for the proposed problems.

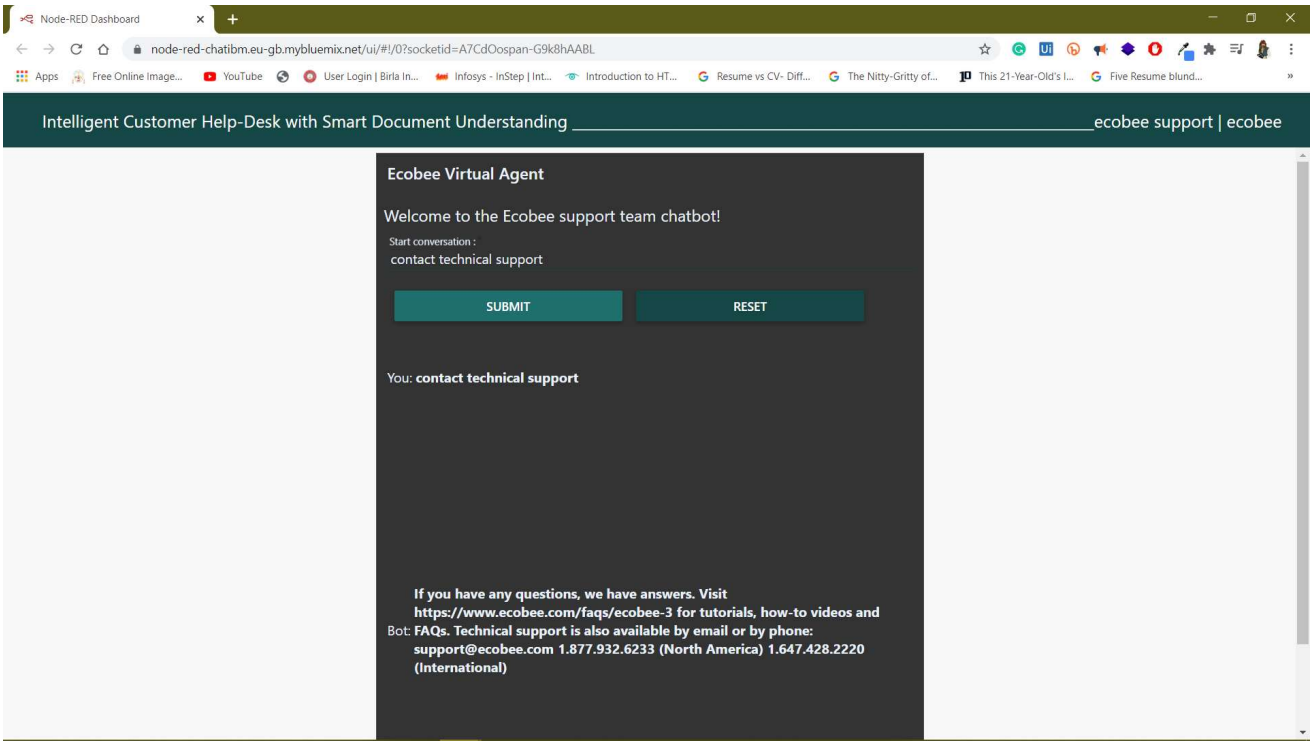


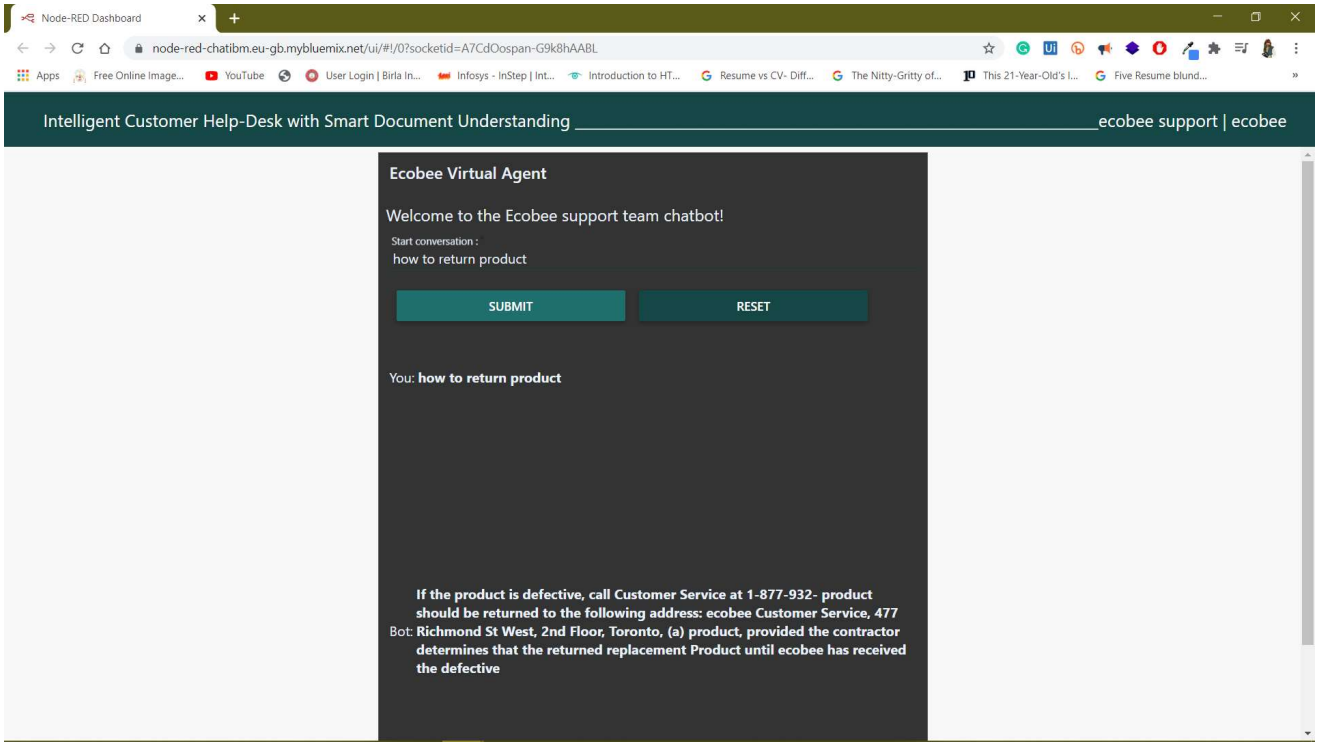
1. The document is annotated using Watson Discovery SDU
2. The user interacts with the backend server via the app UI. The frontend app UI is a chatbot that engages the user in a conversation.
3. Dialog between the user and backend server is coordinated using a Watson Assistant dialog skill.
4. If the 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.

6. RESULT

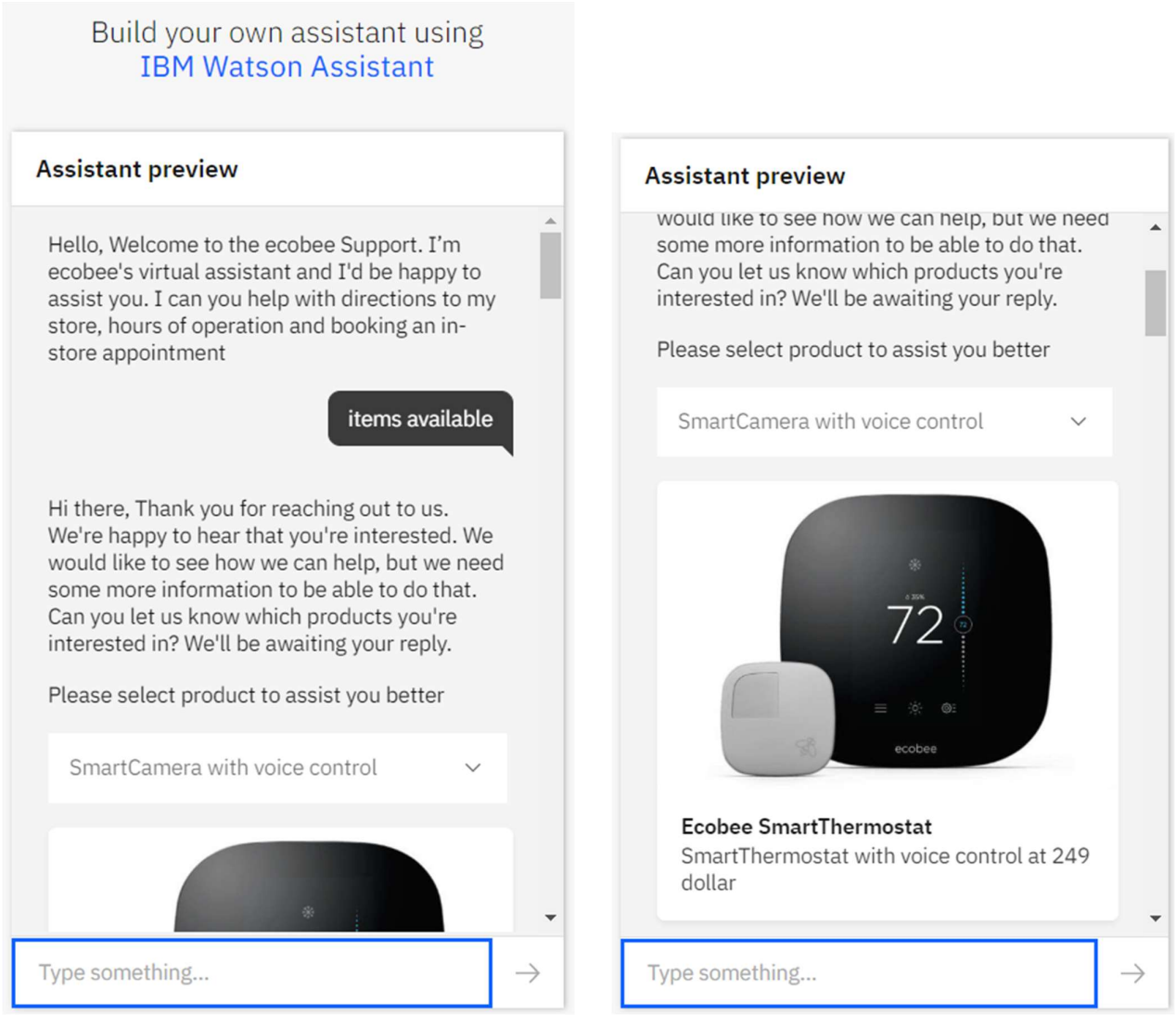
The chatbot was successfully made using Watson assistant and using SDU. All the services were integrated using Node Red Application.

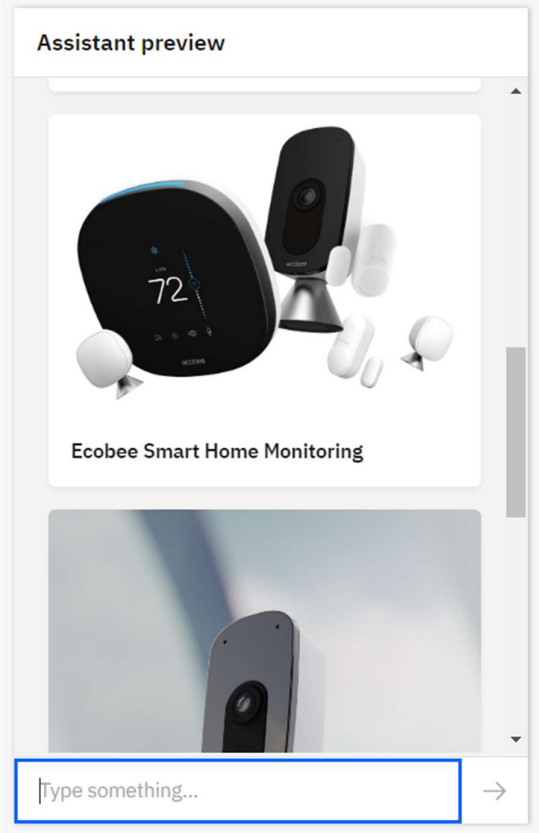


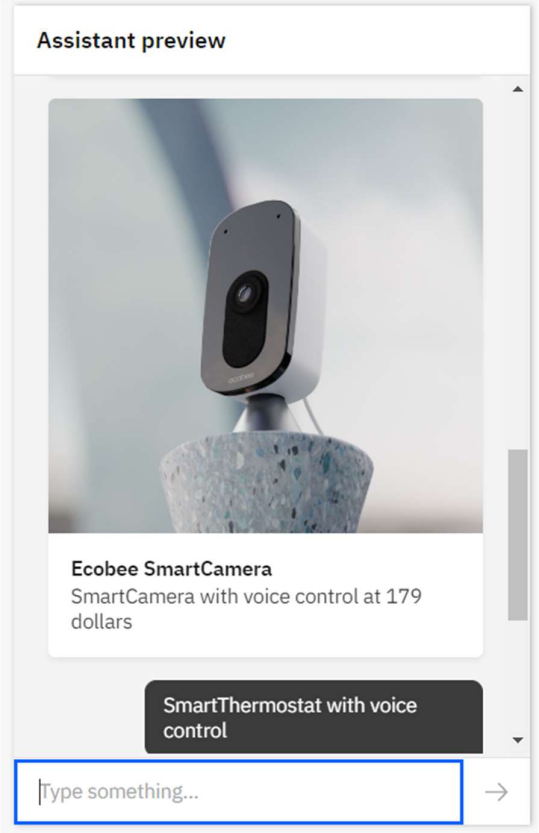




IBM Watson Assistant







7. ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

A) Reduced costs: Chatbots eliminate the need for labor during online interaction with customers. This is obviously a great advantage for companies that receive multiple queries at once. In addition to saving costs with them, companies can align the chatbot with their objectives, and use them as a means to enhance customer conversion.

B) 24/7 Availability: Unlike humans, once we install a chatbot, it can handle queries at any time of day. Thus, the customer does not have to wait for a commercial of the company to help him. This also allows companies to monitor customer « traffic » during non-working hours and contact them later.

C) Learning and updating: AI-based chatbots are able to learn from interactions and update independently. This is one of the main advantages. When you hire a new employee, you have to train them continuously. However, chatbots « form » themselves (with certain limitations, of course).

D) Management of multiple clients: Humans can serve a limited number of customers at the same time. This restriction does not exist for chatbots, and they can manage all the necessary queries simultaneously. This is one of the main advantages of using chatbot, as no customer is left unattended and you are solving different problems at the same time. There are chatbots companies already working on developing voice chatbot services.

DISADVANTAGES:

Complex interface: It is often considered that chatbots are complicated and need a lot of time to understand what you want in customer. Sometimes, it can also annoy the client about their slowness, or their difficulty in filtering responses. They don't get you right: Fixed chatbots can get stuck easily. If a query doesn't relate to something you've previously « taught » it, you won't understand it. This can lead to a frustrated

customer and the loss of the sale. Other times they do understand you, but they need double (or triple) as many messages as one person, which spoils the user experience.

A) Bad memory: The chatbots are not able to memorize a conversation already had, which forces the user to write the same thing over and over again. This can be cumbersome for the client and annoying for the effort required. Therefore, it is important to be careful when designing chatbots and make sure that the program is able to understand users' queries and respond accordingly.

8. APPLICATIONS

Some applications can be: -

1. Help User: This chatbot will be useful for the user to ask the assistant the queries related to the Product and will give them clear guidance about the Product. If the Assistant doesn't know about a certain query, it will redirect to the correct person for it.

2. Content delivery: Media Publishers have realized that chatbots are a powerful way to engage with their audiences and monitor engagement to gain valuable insights on reader interests. Chat with the CNN and Wall Street Journal Chatbots on Facebook Messenger and receive the latest news directly in Messenger, without having to visit their websites.

3. Companionship: The primary function of the chatbot is to be a virtual companion – To speak with senior people on general topics like the weather, nature, hobbies, movies, music, news, etc. The chatbot asks questions, reacts to the answers, is able to speak on various topics, and share interesting news and facts from Google.

9. CONCLUSION

This chatbot will be useful for the user to ask the assistant the queries related to the Product and will give them clear guidance about the product. If the Assistant doesn't know about a certain query, it will redirect to the correct person for it. Chatbots are quickly making transformational changes and allowing businesses to thrive through customer interactions. The feedback and survey through chatbots strengthen the position of businesses as they analyze the reason behind different levels of customer approval. Use of conversational AI chatbots only means better engagement and relentless need for customer satisfaction in the near future.

10. FUTURE SCOPE

Future Scope of this chatbot can be by adding the following to make it more advance: -

1] Smarter Virtual Assistants: Much of what virtual assistants do now are basic skills, such as retrieving data and basic computation. As natural language processing (NLP) continues to mature, virtual assistants will improve their comprehension and response capabilities, allowing for their use to become more widespread and complex. Also, as machine learning progresses, we may see virtual assistants become smarter and begin

to learn and predict customer needs.

2] Integration with IoT Devices: Car speakers, smart home devices, and wearables are just a few examples where the virtual assistant is departing from its original hardware and making its way to in-context devices. These integrations ensure that virtual assistants can always be near their human and ready to support any need. It is expected that these integrations will continue at an accelerated pace throughout 2018.

3] Voice-control: Voice recognition can be added with the virtual assistant. Then the customer can control application by using his voice. Soon, we could be joining meetings with a voice command, instead of dialing in the long meeting ID and password.

11. BIBLIOGRAPHY

- https://cloud.ibm.com/docs/overview?topic=overview-whatis-platform
- https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/
- https://www.ibm.com/cloud/architecture/tutorials/cognitive_discovery
- https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started
- https://nodered.org/
- https://github.com/watson-developer-cloud/node-red-labs
- https://www.youtube.com/embed/r7E1TJ1HtM0
- https://www.youtube.com/watch?v=yAZkeBi5-SY&t=44s
- https://developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chatbot-on-nodered/
- https://github.com/IBM/watson-discovery-sdu-with-assistant
- http://www.iotgyan.com/learning-resource/build-chatbot-using-watson-assistant-tool
- http://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red

APPENDIX:

Source Code

1)Cloud functions code

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