PROJECT REPORT

Intelligent Customer Help Desk with Smart Document Understanding

Category: Artificial Intelligence

Application ID: SPS_APL_20200001204
Project ID:SPS_PRO_99
Internship at smartinterz

Manasi Dhokare
mmdhokare@mitaoe.ac.in

Table of Contents

1.Introduction

1.1 Overview

We will be able to build a chatbot that uses various Watson AI Services
like Watson Discovery, Watson Assistant, Watson Cloud Functions and Node-Red to deliver an effective Web based UI through which we can chat with the assistant.
We will integrate the Watson Discovery service with Watson Assistant using webhooks.

- Project Requirements :Python, Node-RED, IBM Cloud, IBM Watson, NodeJS
- Functional Requirements : IBM Cloud
- Technical Requirements : AI, ML, Watson AI, NodeJS
- Software Requirements : Watson Assistant, Watson Discovery, Watson Cloud Functions, Node-RED
- Project Deliverables : Smartinternz Internship

1.2 Purpose

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 of chatbot, it typically tell the customer the question isn't valid or offer to speak to a real person.
In this project, we will be emphasizing to reduce the work load from the representative by supervising our chatbot.th 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. So unless and untill customer specifically asks for a customer representative the bot will try to solve all your queries.

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.

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 & deploy the same on IBM Cloud Platform.

2. Literature Survey

2.1 Existing Problem

Generally chatbots are loaded wth a certain set of questions that is more like if and else flow, the question or user input which lies out of the scope of the chatbot is not answered and rather a message like "The question isn’t valid"is displayed or offer to speak to the customer agent or the representatives but an efficient chatbot should reduce the traffic reaching to the representitives, So to achieve this we include a Smart chatbot so that it can answer the queries of customer.

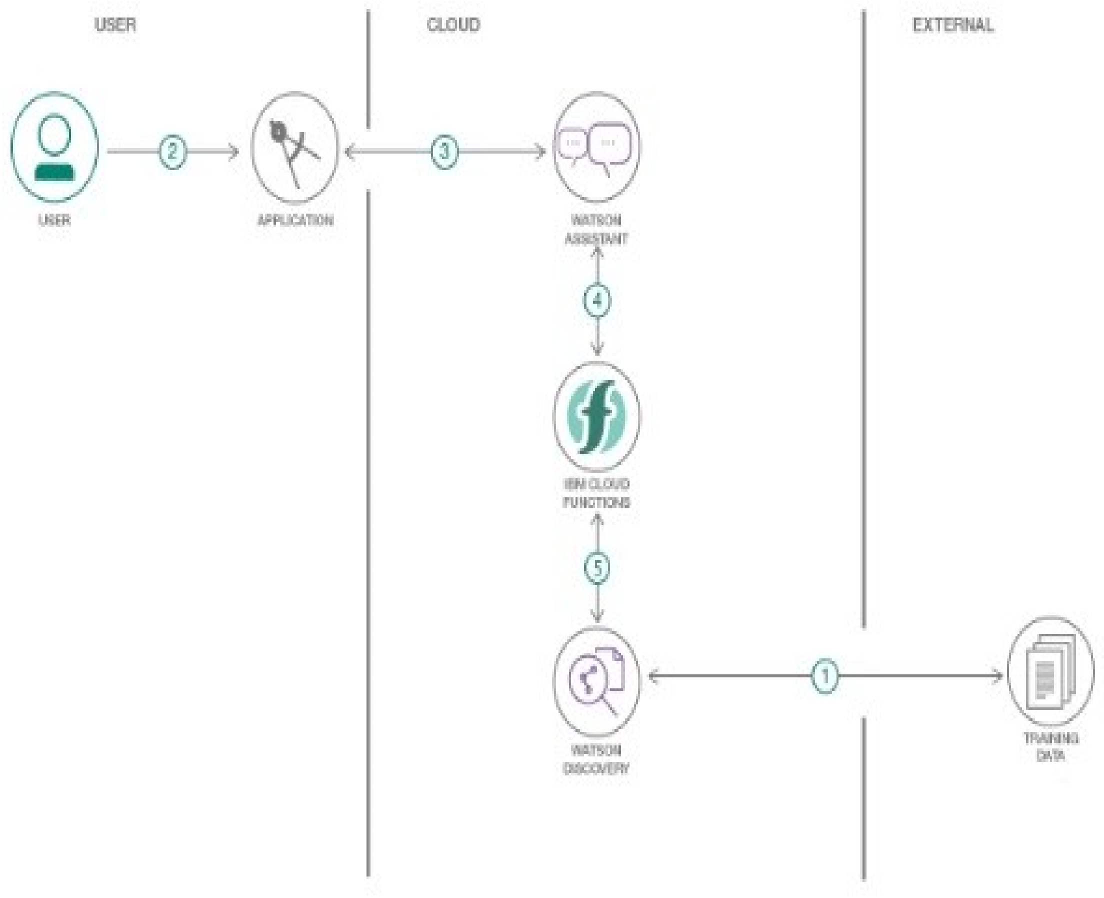
2.2 Proposed Solution

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. So unless and untill customer specifically asks for a customer representative the bot will try to solve all yourqueries.

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. Then using Watson actions as webhook, Watson Discovery can be integrated with Watson assistant. Finally using Node-Red, Watson assistant can be integrated with a web UI. This UI can then be used to connect with Watson assistant and chat with it.

3.Theoretical Analysis

3.1 Block/Flow Diagram



1. The document is annotated using Watson Discovery SDU.
2. The user interacts with the backend server using the application UI. The frontend application UI is a chatbot that engages a user in a conversation.
3. Conversation between user and backend server is coordinated by a watson Assistant dialog skill.
4. If the user asks the product operation question, a search query is passed to a predefined IBM cloud function action.

5. Cloud function action will querry the Watson Discovery service and return the results.

3.2 Hardware/Software Designing

- Create necessary IBM Cloud Services.
- Configure Watson Discovery.
- Create IBM Cloud Function Action.
- Configure Watson Assistant.
- Integrate Watson Discovery with Watson Assistant using webhook.
- Build   Node-RED flow to integrate Watson Assistant and Web Dashboard.

4. Experimental Investigation

1. Create IBM Cloud Services

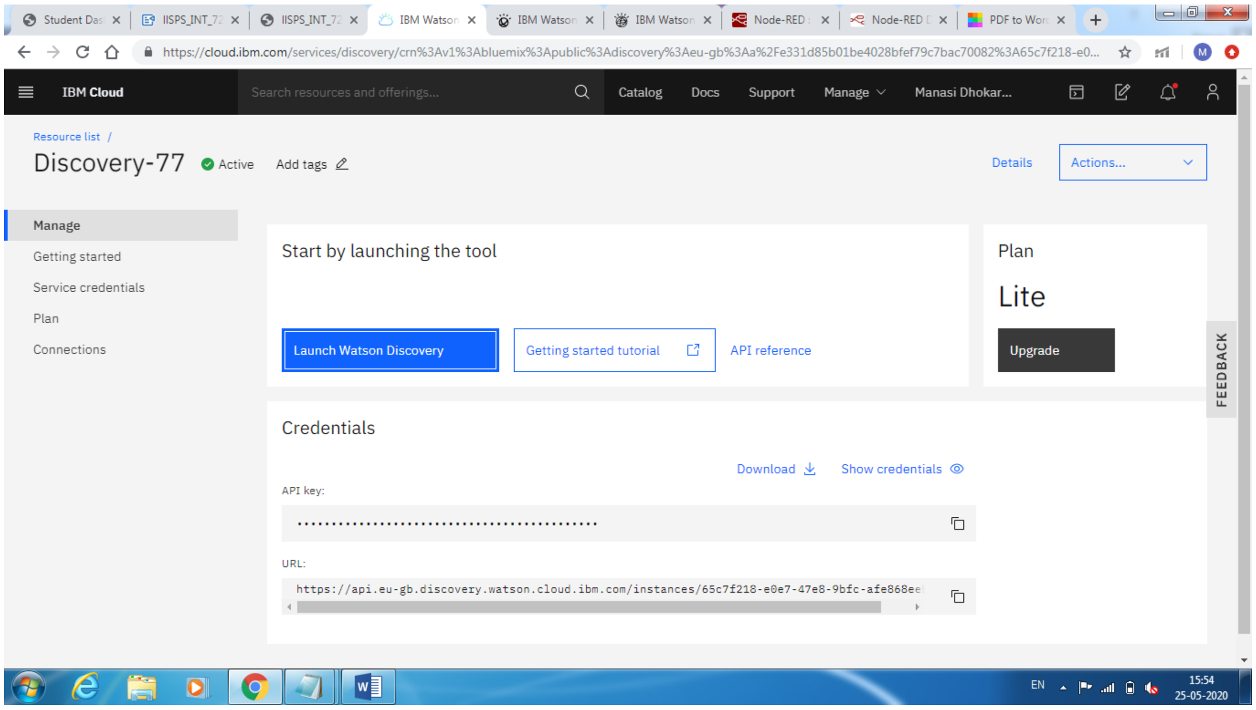
a. Watson Discovery
b. Watson Assistant
c. Node Red

2. Configure Watson Discovery

After creating and launching discovery, Import the document on which we need to train the discovery service.We have selected the ecobee3 user guide located in the data directory of our local repository.
This can be done easily by clicking on the configure setting and then labeling each word or element present in the document as their respective label such as title, subtitle, text, image,and footer. some of the labels are not present in the lite plan.

Steps:

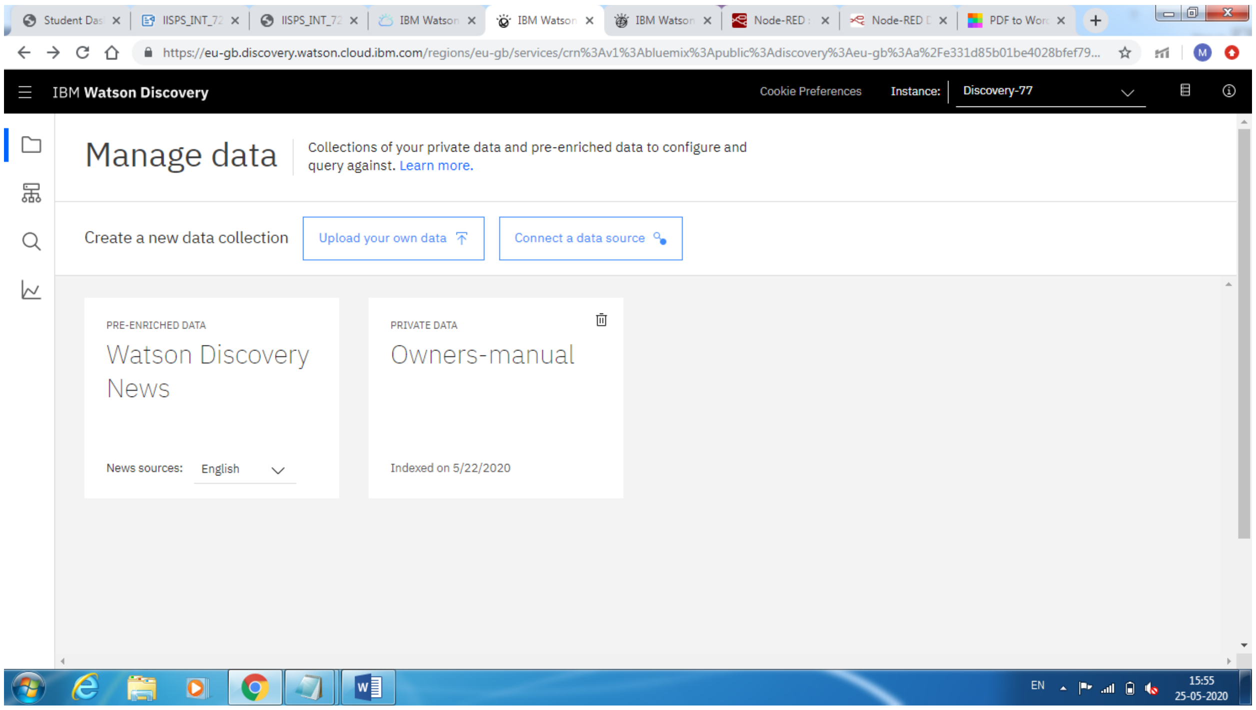
After creating the discovery, we will redirected to this page of discovery where there is name of the discovery along with API key and URL.



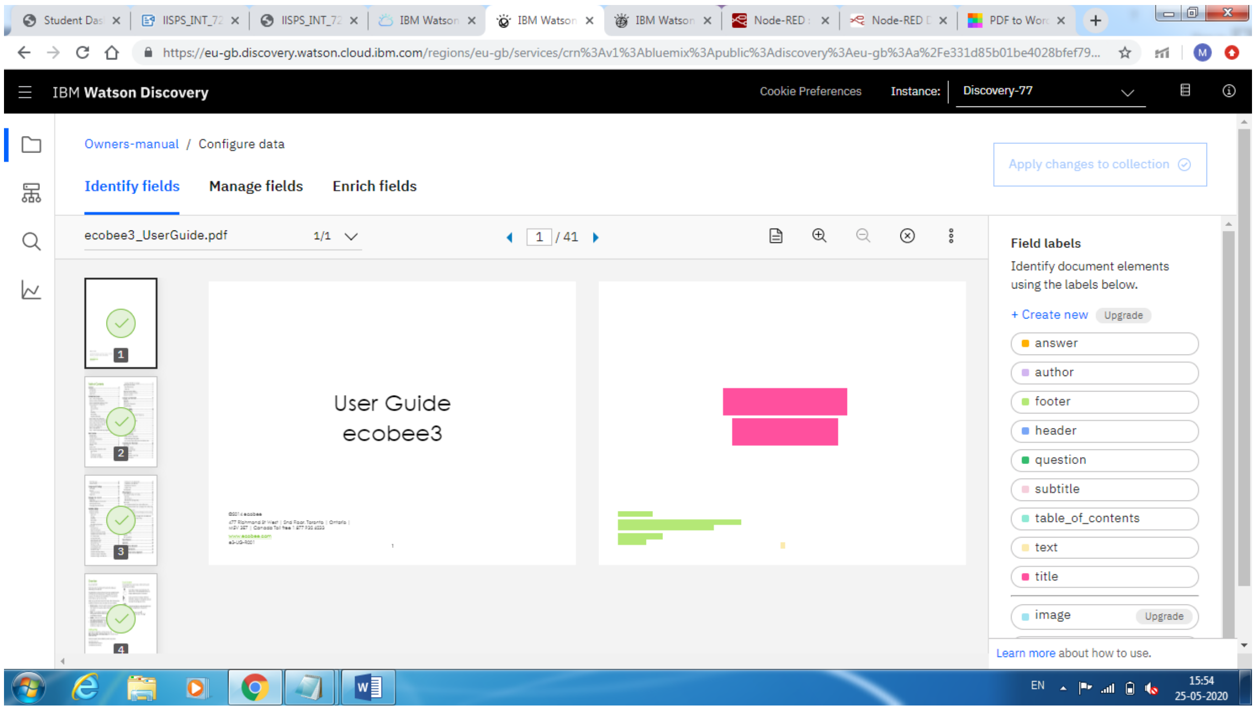
API key: ••

URL: https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/65c7f218-e0e7-47e8-9bfc-afe868ee

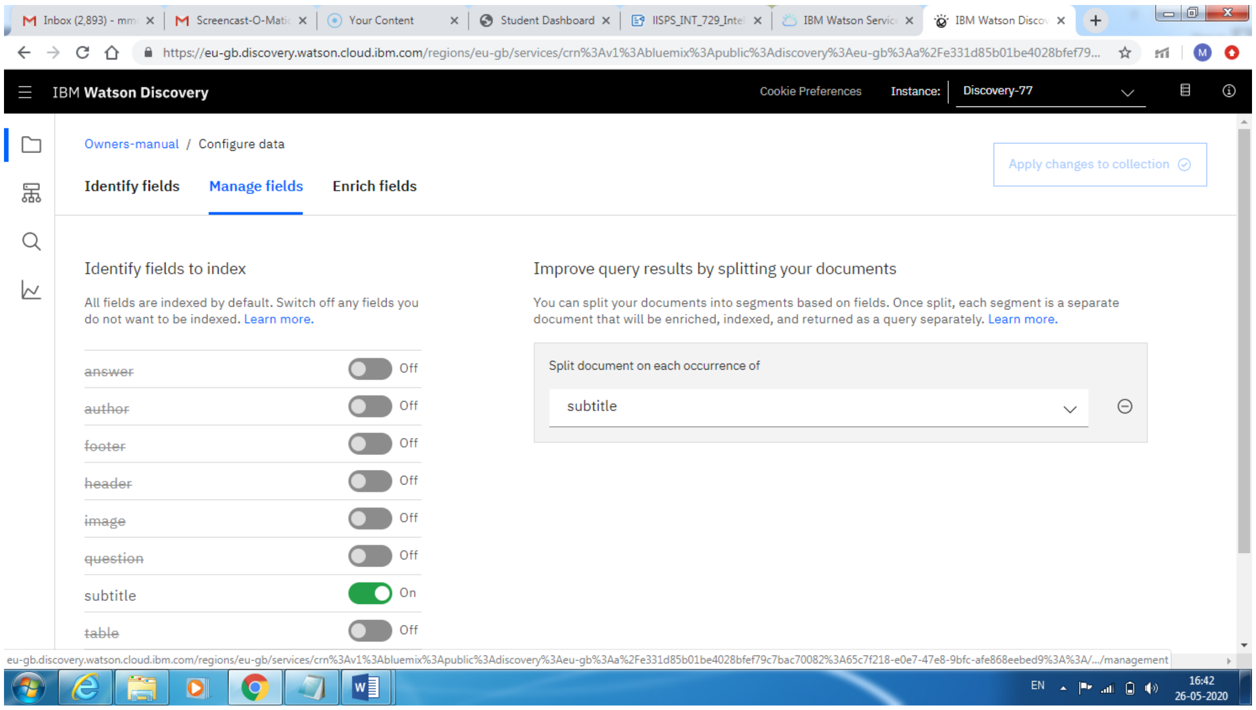
- After launching discovery next step we have to upload the data by clivking on upload data.



- Below is the layout of identified fields tab of the SDU annotation panel.



The aim is to annotate all the pages of the document, so that discovery can learn what ttext is important and what text can be ignored.
From each page we have to train discovery that the labels of pages like header, footer, text, subtitle, etc.
For further segmentation and making the sub documents, we have to manage the fields, Here we are provided with the option of identifying fields to index i.e what all texts are important for us.



Identify fields to index

All fields are indexed by default. Switch off any fields you do not want to be indexed. Learn more.

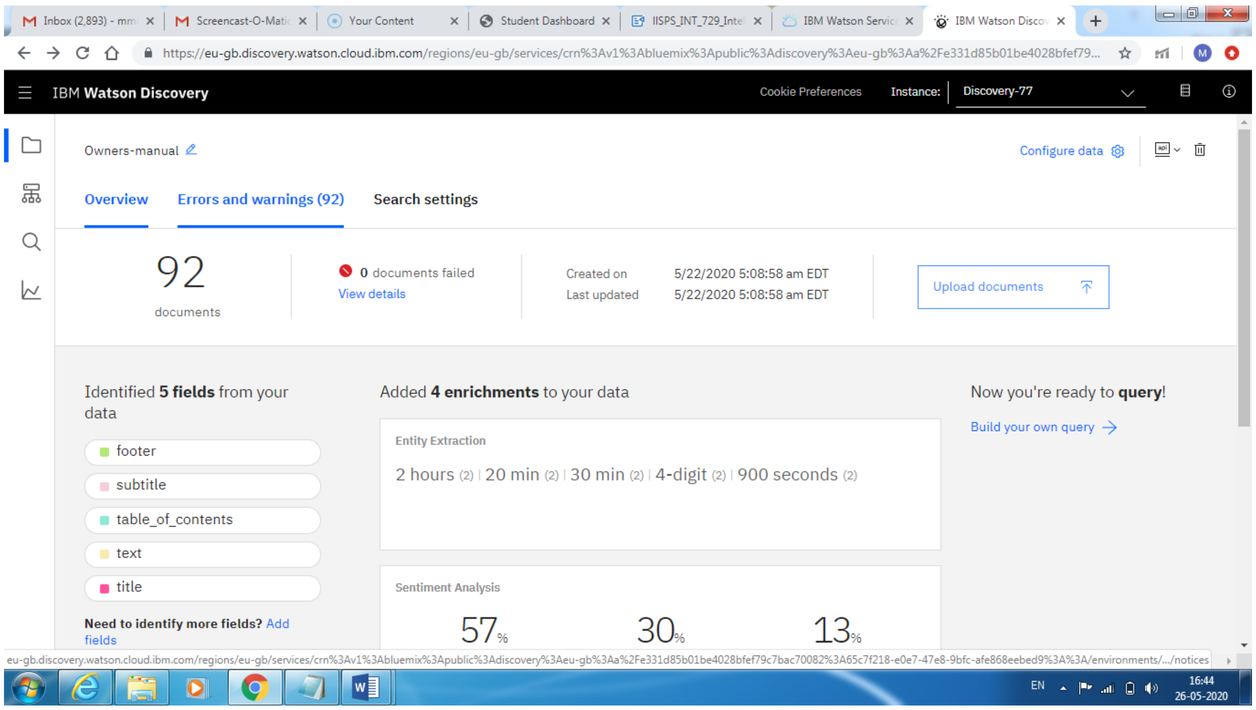
~~answer~~	Off
~~author~~	Off
~~footer~~	Off
~~header~~	Off
~~image~~	Off
~~question~~	Off
subtitle	On
~~table~~	Off

Improve query results by splitting your documents

You can split your documents into segments based on fields. Once split, each segment is a separate document that will be enriched, indexed, and returned as a query separately. Learn more.

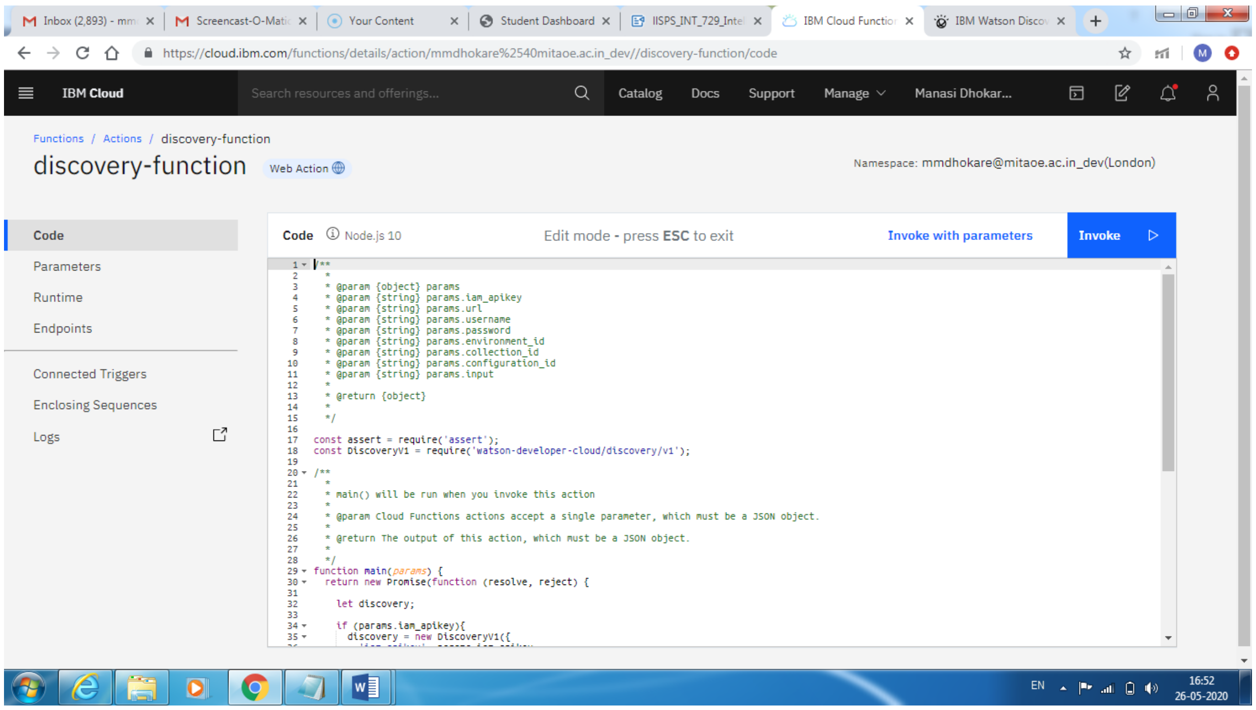
Split document on each occurrence of: subtitle

After doing this step we get 92 documents as shown in below.



3. Cloud Function Action

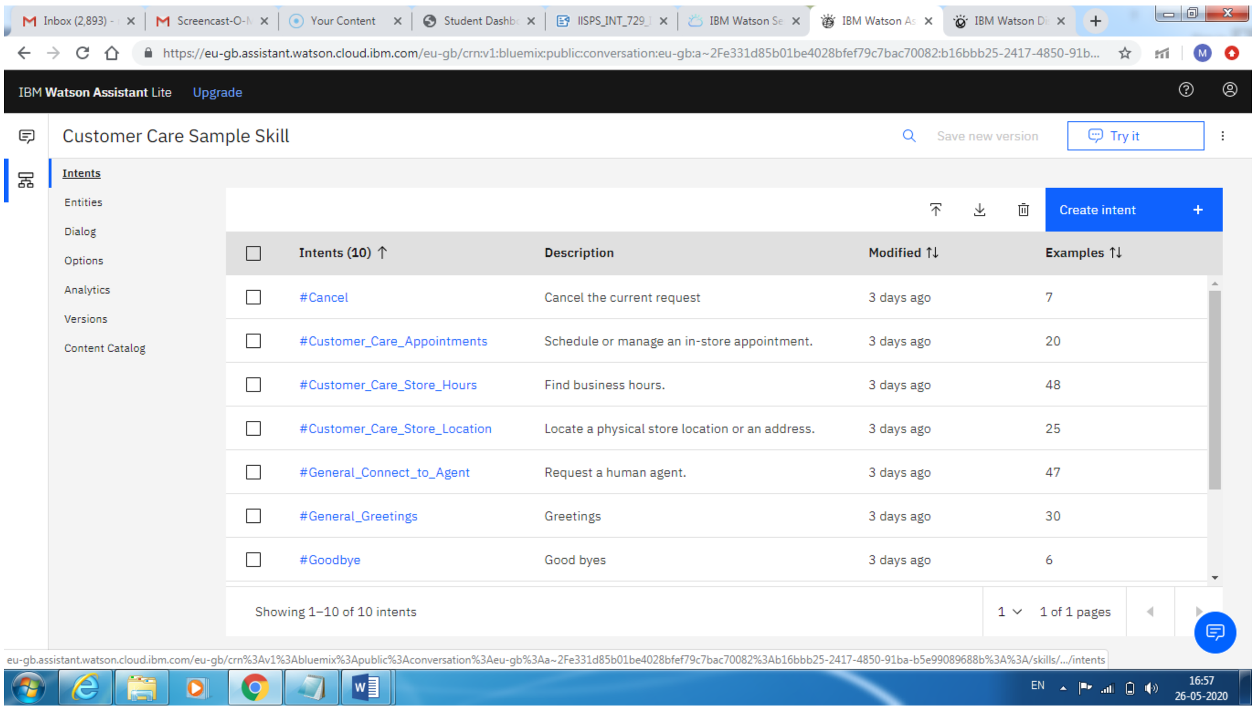
Next we have to create the IBM Cloud Function Action:
It is used to link the Watson Discovery with Watson Assistant so that our querries can handled by discovery.



```
/**
  *
  * @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({
```

4. Configure Watson Assistant

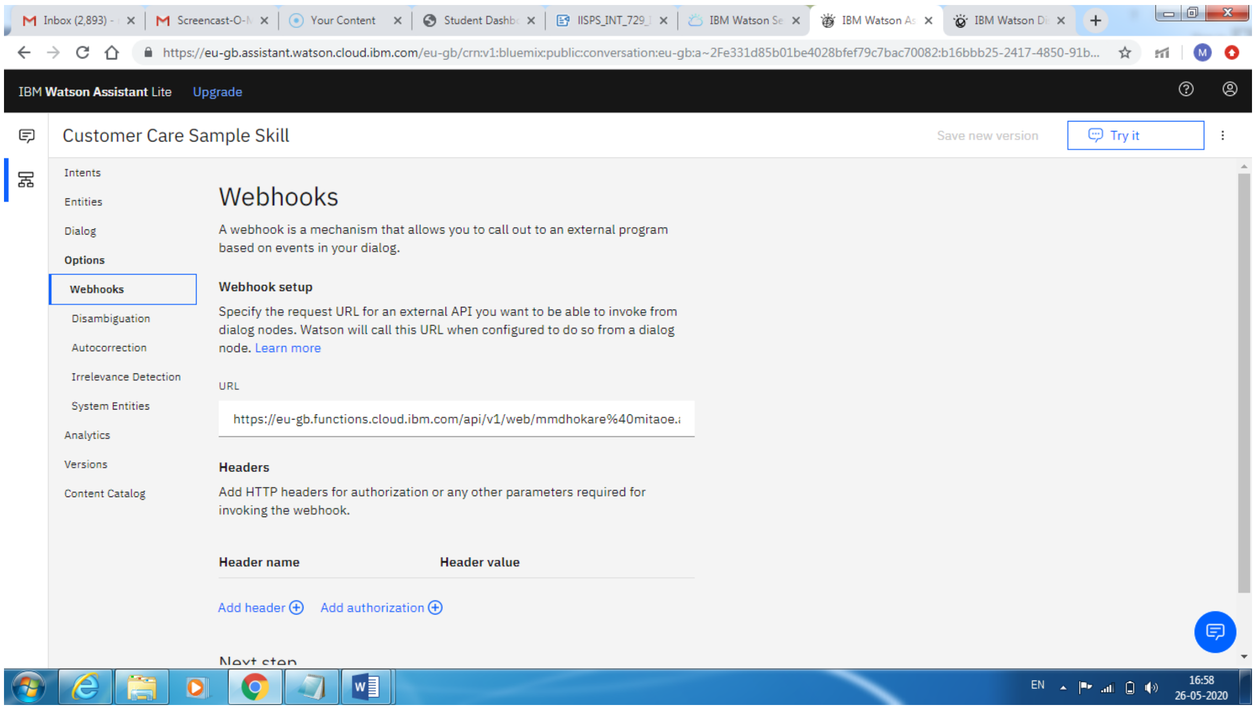
Next, we have to create a watson assistant and create a customer care skill. We can create intent , entities and dialogue related to our skill.



	Intents (10)	Description	Modified	Examples
	#Cancel	Cancel the current request	3 days ago	7
	#Customer_Care_Appointments	Schedule or manage an in-store appointment.	3 days ago	20
	#Customer_Care_Store_Hours	Find business hours.	3 days ago	48
	#Customer_Care_Store_Location	Locate a physical store location or an address.	3 days ago	25
	#General_Connect_to_Agent	Request a human agent.	3 days ago	47
	#General_Greetings	Greetings	3 days ago	30
	#Goodbye	Good byes	3 days ago	6

Showing 1–10 of 10 intents   1 of 1 pages

We have to enable the webhooks which enables dialog to send a POST request to the webhook URL.



Webhooks

A webhook is a mechanism that allows you to call out to an external program based on events in your dialog.

Webhook setup

Specify the request URL for an external API you want to be able to invoke from dialog nodes. Watson will call this URL when configured to do so from a dialog node. Learn more

URL

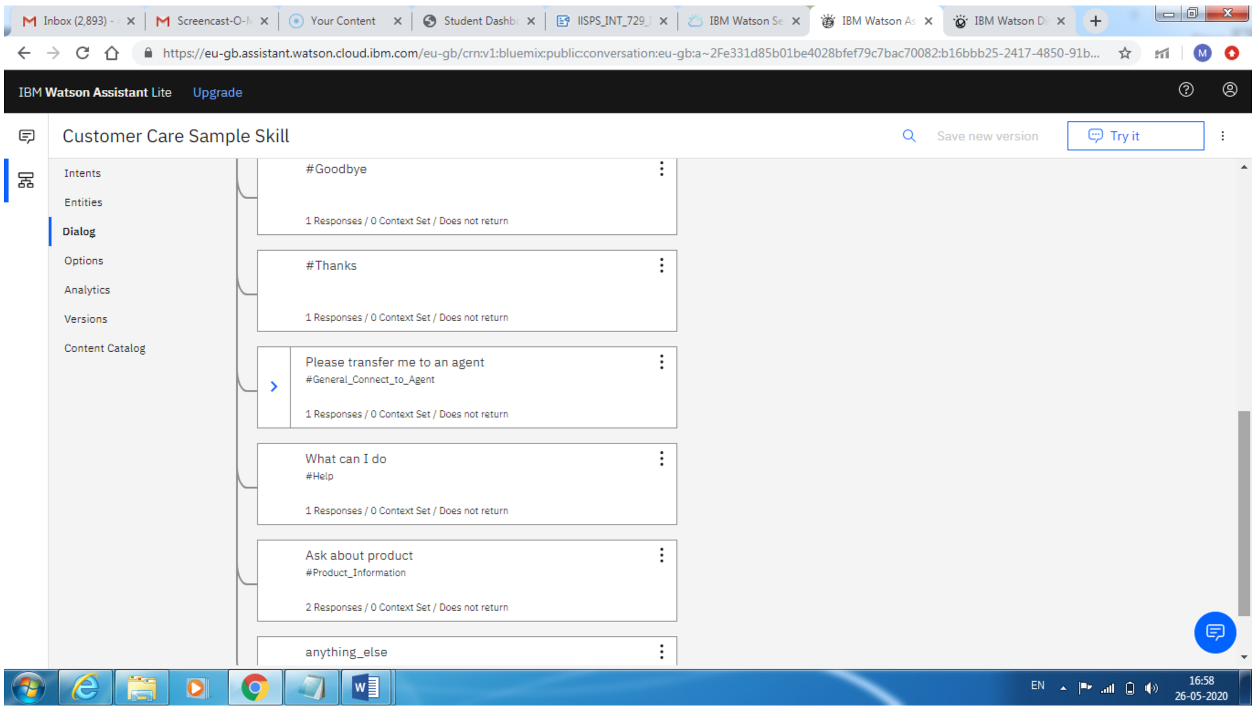
https://eu-gb.functions.cloud.ibm.com/api/v1/web/mmdhokare%40mitaoe.i

Headers

Add HTTP headers for authorization or any other parameters required for invoking the webhook.

Header name	Header value

Add header ⊕ Add authorization ⊕



#Goodbye
1 Responses / 0 Context Set / Does not return

#Thanks
1 Responses / 0 Context Set / Does not return

Please transfer me to an agent
#General_Connect_to_Agent
1 Responses / 0 Context Set / Does not return

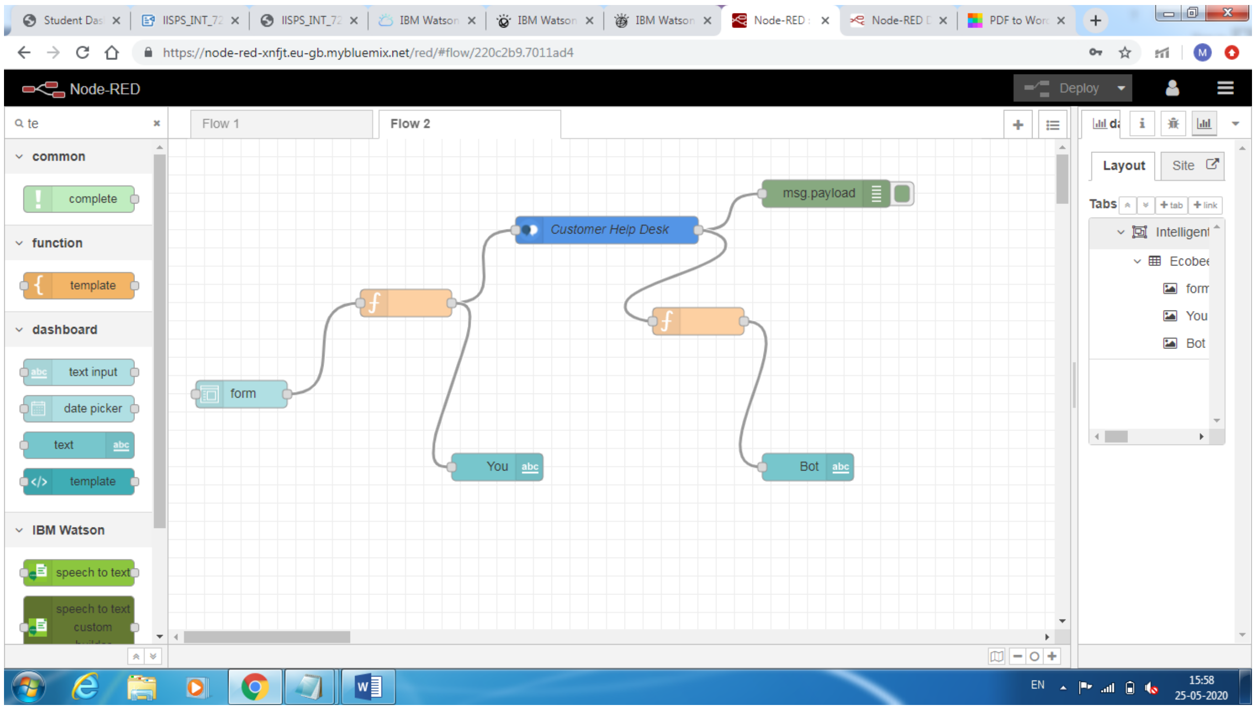
What can I do
#Help
1 Responses / 0 Context Set / Does not return

Ask about product
#Product_Information
2 Responses / 0 Context Set / Does not return

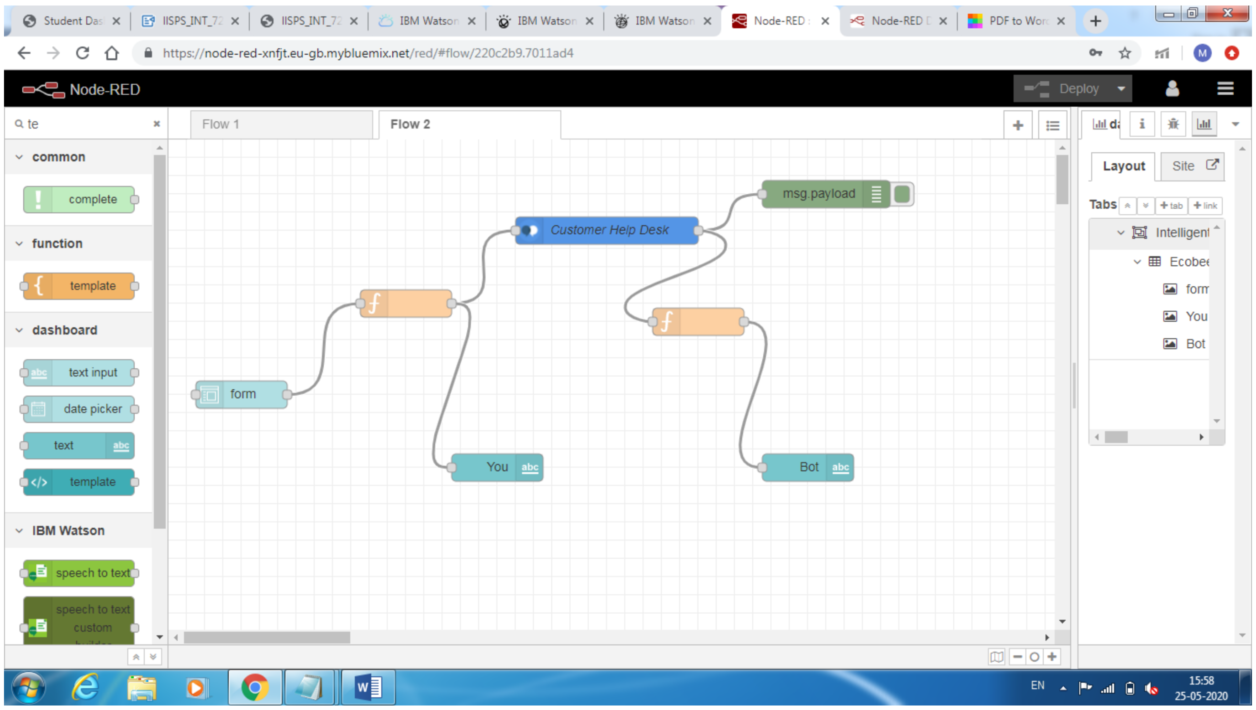
anything_else

5. Node-Red Flow

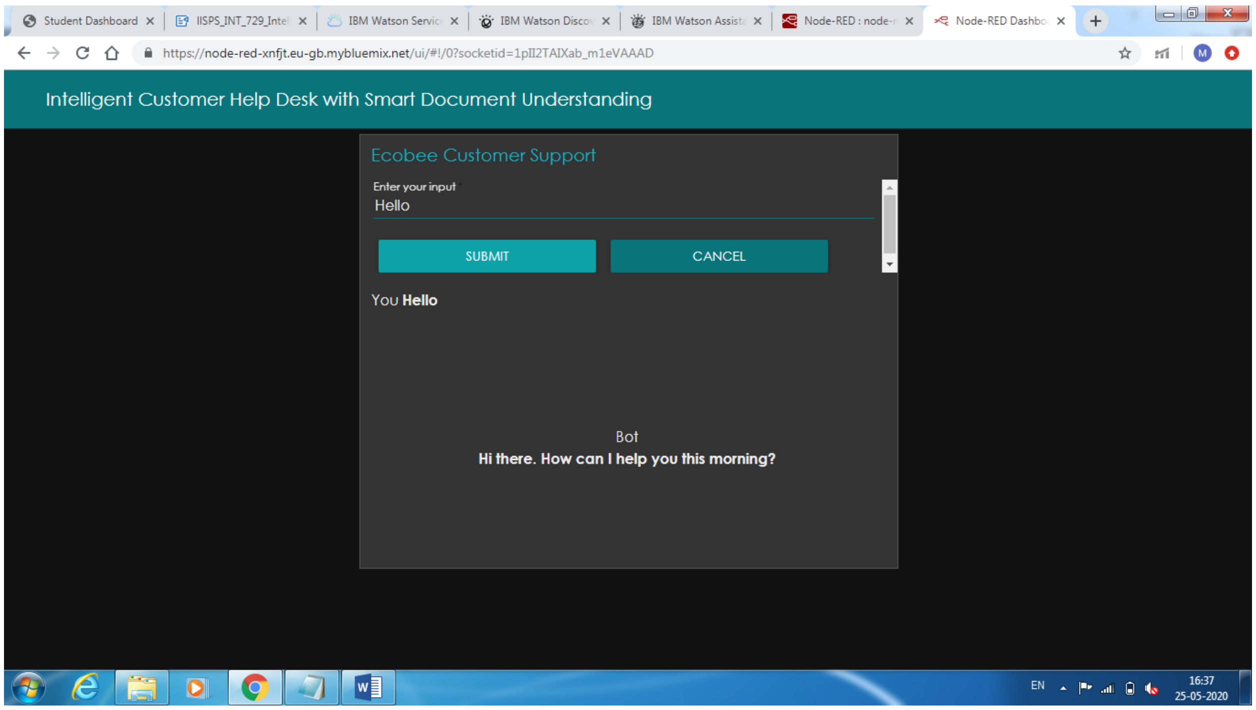
After this we have to make Node-Red flow and link all things. we will get UI from the node.



5. Flowchart


Node-RED
Deploy
Flow 1
Flow 2
common
complete
function
template
dashboard
text input
date picker
text
template
IBM Watson
speech to text
speech to text custom
msg.payload
Customer Help Desk
form
You
Bot
Layout
Site
Tabs
Intelligent
Ecobee
form
You
Bot

6. Result



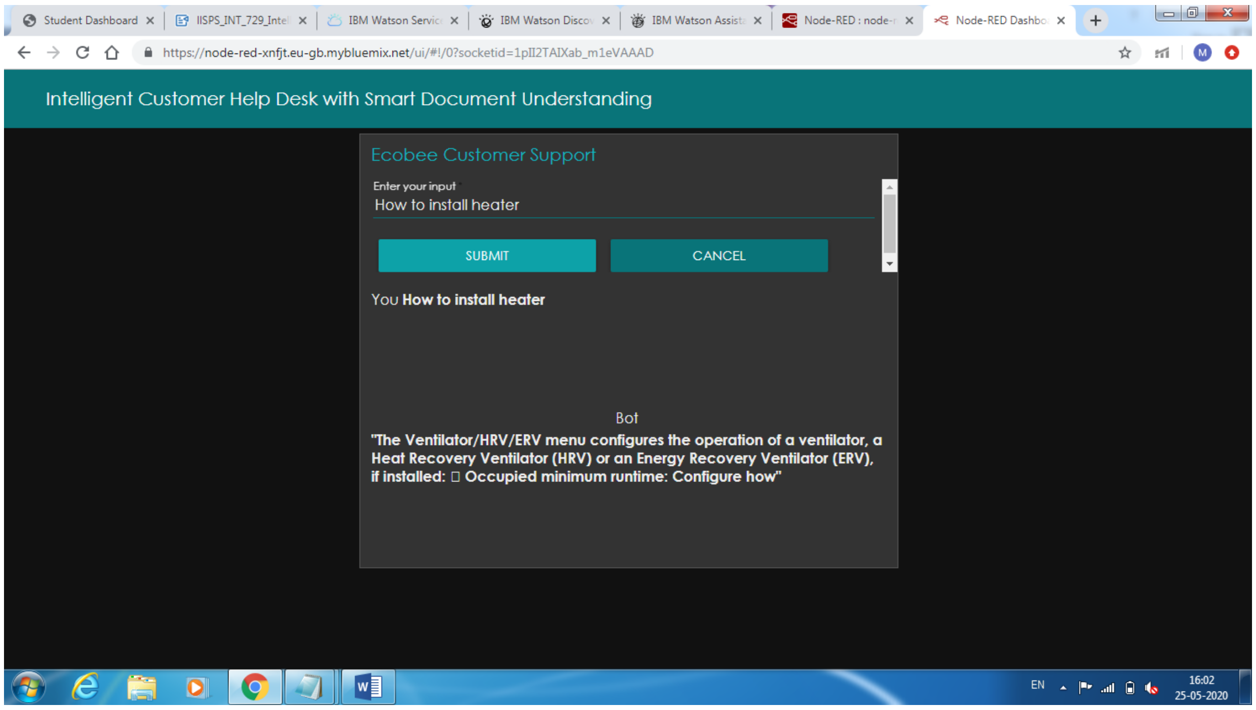
Intelligent Customer Help Desk with Smart Document Understanding

Ecobee Customer Support

Enter your input
Hello

You **Hello**

Bot
Hi there. How can I help you this morning?



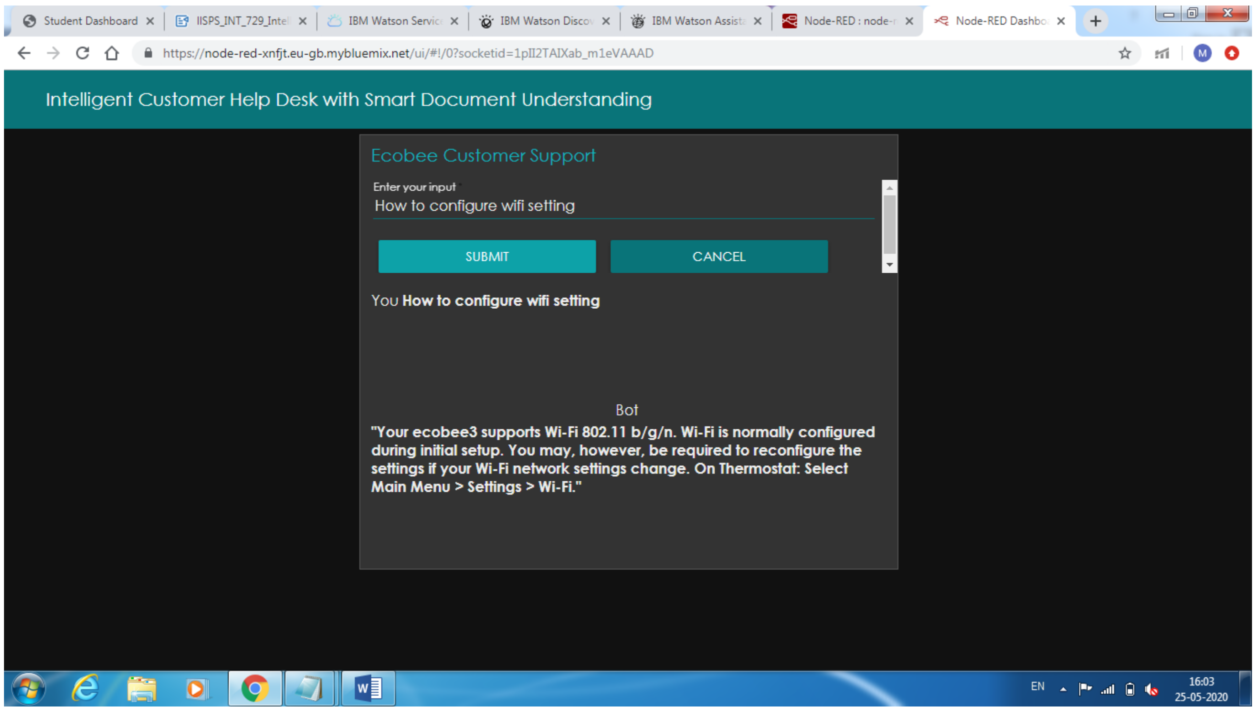
Intelligent Customer Help Desk with Smart Document Understanding

Ecobee Customer Support

Enter your input
How to install heater

You **How to install heater**

Bot
"The Ventilator/HRV/ERV menu configures the operation of a ventilator, a Heat Recovery Ventilator (HRV) or an Energy Recovery Ventilator (ERV), if installed: □ Occupied minimum runtime: Configure how"



Intelligent Customer Help Desk with Smart Document Understanding

Ecobee Customer Support

Enter your input
How to configure wifi setting

You **How to configure wifi setting**

Bot

"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 and Disadvantages

7.1 Advantages:

- Companies can use these to decrease the work flow to the representatives.
- Reduces Man Power.
- Cost Efficient.
- Decrease in the number of calls diverted to representatives.
- Less work load on employees.

7.2 Disadvantages:

- Sometime the chatbot misleads the customer
- The discovery returns wrong results when not properly configured.
- Giving same answer for different sentiments.

8. Application

- This chatbot can be deployed to various websites like facebook, slack, telegram etc. as it can solve a lot of basic questions.
- It can be used to deploy as Customer Helpdesk for small scale productsas their manual usually has the solution for the user'sproblems.
- Chatbot can be deployed at any website to clear the basic doubts of customer.

9. Conclusion

I have succesfully implemented a project by creating a basic chatbot which can help us to answer the basic questions of the customer or user related to location of office, working hours and the information about the product. We successfully create the intelligent helpdesk smart chatbot using Watson Assistant, Watson Cloud Function, Watson Discovery and Node-Red.

10. Future Scope

We can improve the results of discovery by enriching it with more fields and doing the Smart Data Annotation more accurately. We canget premium version to increase the scope of our chatbot. We can import pre-built node-red flow and can improve our UI. Also we can make database to store our chats and use it show recent chats to customer.
We can also include other IBM Watson functinalities like text to seech and speech to text services to access chatbot handfree.

11. Bibilography

1. Node-RED Starter Application: https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/
2. Build your own AI assistant : https://www.youtube.com/watch?v=hitUOFNne14
3. How to use Watson Assistant with Webhooks : https://www.youtube.com/embed/5z3i5IsBVnk
4. Watson Discovery : https://developer.ibm.com/articles/introduction-watson-discovery/

12. Appendix

Code:

Cloud Function

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

Node-Red flow code

```
[
        {
            "id": "220c2b9.7011ad4",
            "type": "tab",
            "label": "Flow 2",
            "disabled": false,
            "info": ""
        },
        {
            "id": "c074f185.5f021",
            "type": "ui_form",
            "z": "220c2b9.7011ad4",
            "name": "",
            "label": "",
            "group": "58c3ba04.e23344",
            "order": 1,
            "width": 0,
            "height": 0,
            "options": [
                {
                    "label": "Enter your input",
                    "value": "text",
                    "type": "text",
                "required": true,
                "rows": null
            }
        ],
        "formValue": {
            "text": ""
        },
        "payload": "",
        "submit": "submit",
        "cancel": "cancel",
        "topic": "",
        "x": 80,
        "y": 280,
        "wires": [
            [
                "cc6b20a3.e75fc"
            ]
        ]
    },
    {
        "id": "cc6b20a3.e75fc",
        "type": "function",
        "z": "220c2b9.7011ad4",
        "name": "",
        "func": "msg.payload=msg.payload.text;\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 260,
        "y": 180,
        "wires": [
            [
                "ed3a5b7c.eb74d8",
                "bf590b69.4f6bf8"
            ]
        ]
    },
    {
        "id": "999d0042.a0ad8",
        "type": "function",
        "z": "220c2b9.7011ad4",
        "name": "",
        "func": "msg.payload=msg.payload.output.text[0];\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 580,
        "y": 200,
        "wires": [
            [
                "f58527ac.d8e8e8"
            ]
        ]
    },
    {
        "id": "ed3a5b7c.eb74d8",
        "type": "watson-conversation-v1",
        "z": "220c2b9.7011ad4",
        "name": "Customer Help Desk",
        "workspaceid": "44fea49a-06c3-4170-b16d-ea4c70e63e7d",
        "multiuser": false,
        "context": true,
        "empty-payload": false,
        "service-endpoint":
"https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/b16bbb25-2417-4850-
91ba-b5e99089688b",
        "timeout": "",
        "optout-learning": false,
        "x": 480,
        "y": 100,
        "wires": [
            [
                "79ff4413.49a6bc",
                "999d0042.a0ad8"
            ]
        ]
    },
{
    "id": "bf590b69.4f6bf8",
    "type": "ui_text",
    "z": "220c2b9.7011ad4",
    "group": "58c3ba04.e23344",
    "order": 2,
    "width": 0,
    "height": 0,
    "name": "",
    "label": "You",
    "format": "{{msg.payload}}",
    "layout": "row-left",
    "x": 360,
    "y": 360,
    "wires": []
},
{
    "id": "79ff4413.49a6bc",
    "type": "debug",
    "z": "220c2b9.7011ad4",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "false",
    "x": 720,
    "y": 60,
    "wires": []
},
{
    "id": "f58527ac.d8e8e8",
    "type": "ui_text",
    "z": "220c2b9.7011ad4",
    "group": "58c3ba04.e23344",
    "order": 3,
    "width": 11,
        "height": 5,
        "name": "",
        "label": "Bot",
        "format": "{{msg.payload}}",
        "layout": "col-center",
        "x": 700,
        "y": 360,
        "wires": []
    },
    {
        "id": "58c3ba04.e23344",
        "type": "ui_group",
        "z": "",
        "name": "ChatBot",
        "tab": "cab09e34.0ed41",
        "order": 1,
        "disp": true,
        "width": 11,
        "collapse": false
    },
    {
        "id": "cab09e34.0ed41",
        "type": "ui_tab",
        "z": "",
        "name": "Customer Helpdesk",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    }
]
```