

Project Report

Project Topic : - Intelligent Customer Help Desk with Smart Document Understanding.

Category : - Machine learning Engineer

Name : - Shivendra Patel

Email ID : - Shivendra123patel2gmail.com

Internship : - Smart Bridge (Remote Summer Internship Program 2020)

Website : - <https://smartinternz.com/>

CONTENTS

1. Introduction

1.1. Overview

1.2. Purpose

1.2.1. Scope of Work

2. Literature Survey

2.1. Existing problem

2.2. Proposed Solution

3. Theoretical Analysis

3.1. Block Diagram

3.2. Hardware/Software Design

4. Experimental Investigations

5. Flowchart

6. Result

7. Advantages & Disadvantages

8. Application

9. Conclusion

10. Future Scope

11. Appendix

11.1. Source Code

11.2. References

1. Introduction

1.1 Overview

In this project we will be able to write an application with the help of IBM Cloud Services (Watson AI services like Discovery, Assistant, Cloud Function and Node Red). By the end of the project we will get a Chatbot which act as an intelligent web based Customer support Chatbot or after sales Service and Customer queries.

- Project Requirements: IBM Cloud account, Python, Node Red, IBM Watson Services.
- Functional Requirements: IBM Cloud Services
- Technical Requirements: Python, Node JS, Watson AI, ML
- Software Requirements: Watson Assistant, Watson Discovery, IBM Cloud Function, Node Red
- Project Deliverables: Smartinternz Internship
- Project Duration: 1 Month

1.2 Purpose

When we purchase any electronic machine then a customer care will explain everything about the electronic goods. Basically Chatbot is just like a customer care which usually can answer just simple questions, such as store locations and hours, directions and maybe even making appointments. When any user input (question) falls out of the scope of the predetermined question set of Chatbot, it typically tells us to rephrase our sentence or mention that this question is not valid.

In this project, there is another option. If the customers' question is about the operation of a device, the application passes the question onto the IBM Watson Discovery Service, which has been preloaded with the device's owners manual. So now, instead of "Would you like to speak to a customer representative?" we return relevant sections of the owner's manual to help solve our customers' problems.

To take it a step further, the project uses the Smart Document Understanding feature of IBM Watson Discovery Service to train it on what text in the owner's manual is important and what is not. This improves 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.

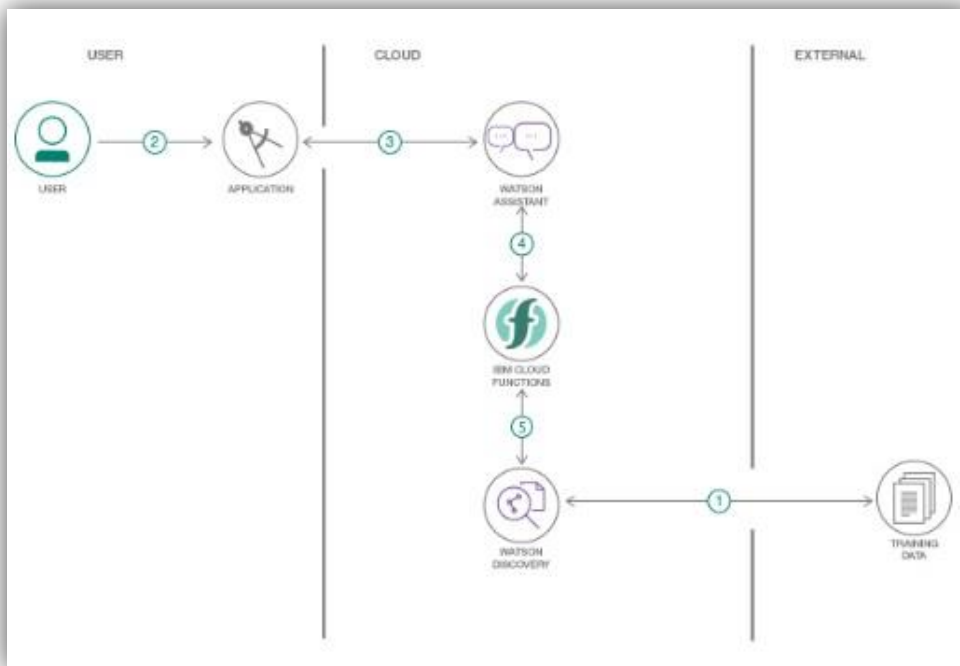
2.1 Existing Problem

Generally Chatbot are loaded with a certain set of questions that is more like if and else flow, the question or the user input which lies out of the scope of the chatbot is not answered and rather a message like “Try again after rephrasing” & “I am unable to understand, Please Rephrase” are displayed and it directs the user to the customer agent or the representative but an efficient chatbot should reduce the traffic reaching to the representatives, So to achieve this we include a Smart chatbot so that it can answer the queries of the customer.

2.2 Proposed Solution

In order to solve the above-mentioned problem, we have to put a virtual agent in chatbot so it can understand the queries that are posted by customers. The virtual agent should be trained from some insight based on company backgrounds, working hours, store locations and product related information. In this project I used Watson Discovery to achieve the above solution.

3.1. Block Diagram



- The document is annotated using Watson Discovery SDU.
- 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.
- Dialog between the user and backend server is coordinated using a Watson Assistant dialog skill.
- If the user asks the product operation question, a search query is passed to a predefined IBM Cloud Function action.
- Cloud function action will query the Watson Discovery service and return the results.

3.2. Hardware/Software Designing

- Create necessary IBM Cloud services.
- Configure IBM Watson Discovery
- Create IBM Cloud Function action
- Configure IBM Watson Assistant
- Create Node-RED flow and configure node
- Deploy and run node red app.

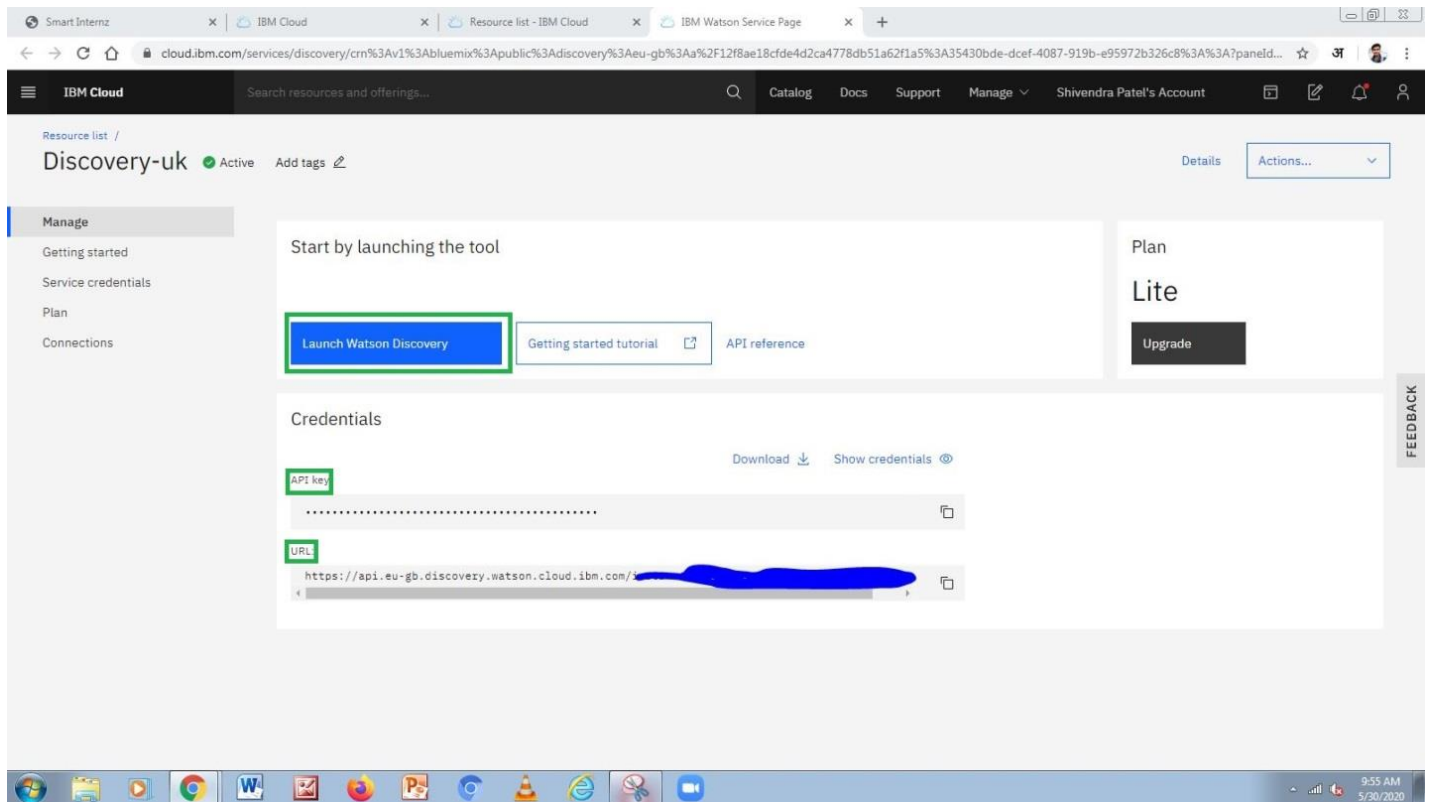
4. Experimental Investigation

1. Create required IBM Cloud Services

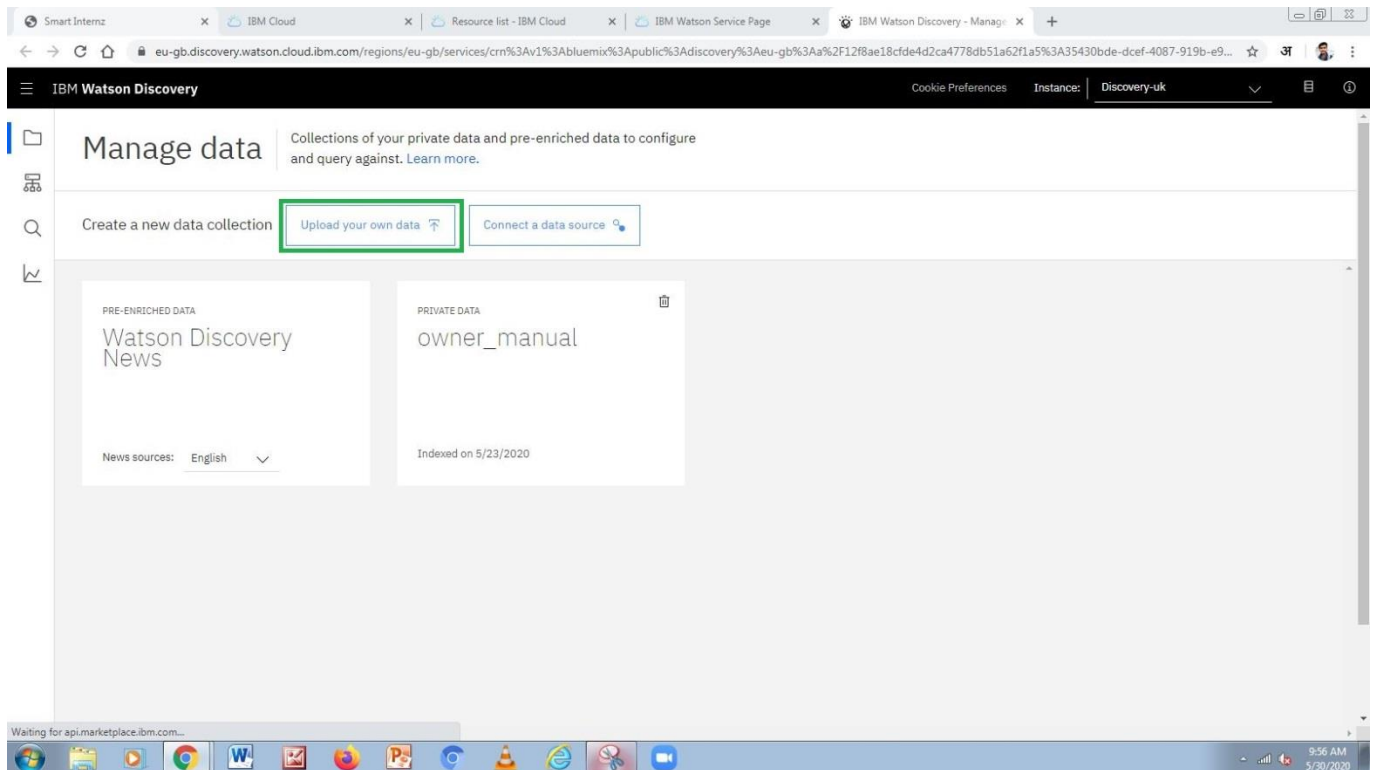
- IBM Watson Assistant
- IBM Watson Discovery
- Node-RED

2. Configure IBM Watson Discovery

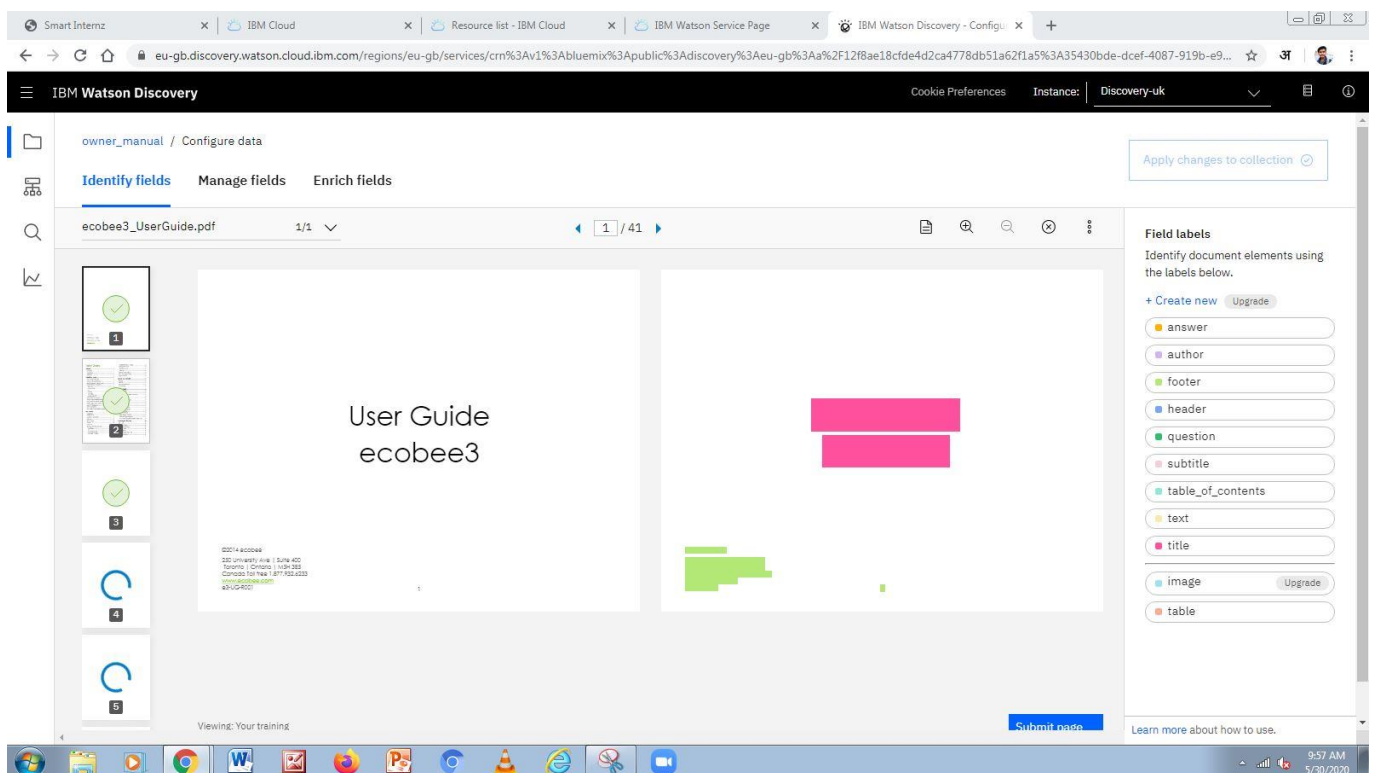
a. Launch Watson Discovery service.



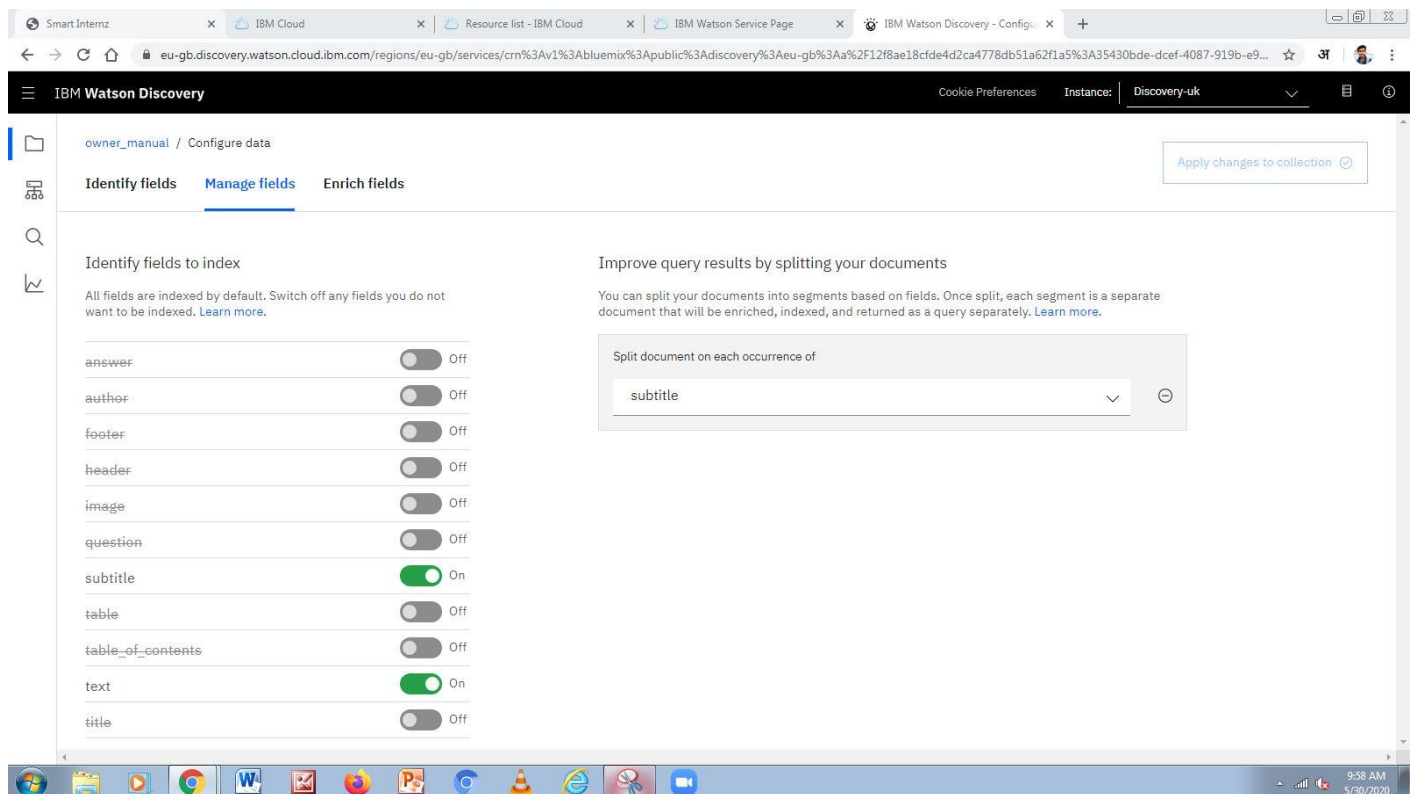
b. Upload documents for Smart Document Understanding (I have used the Ecobee3 Smart Thermostat User Manual).



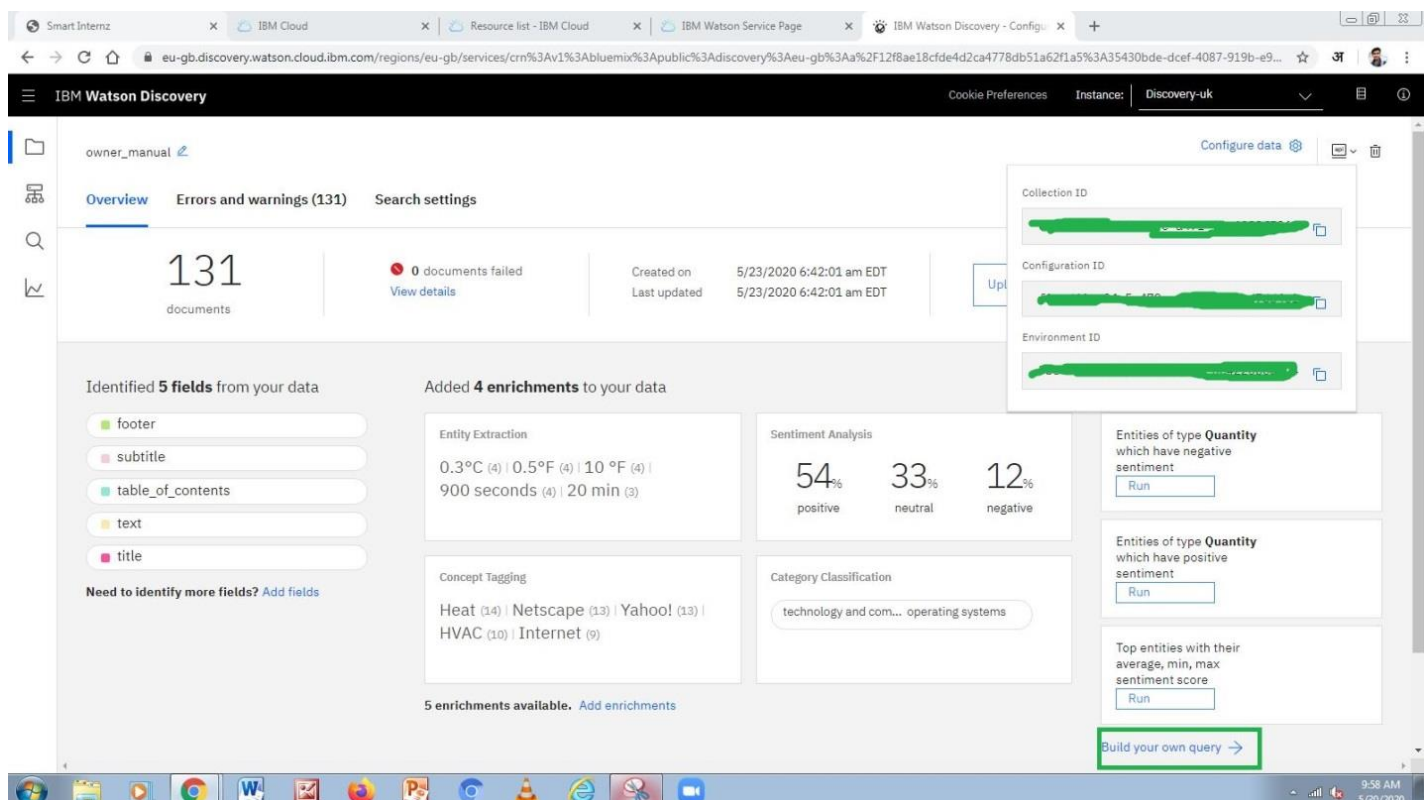
c. Annotate data for Smart Document Understanding.



d. Select the fields to be indexed and split document for improved results (I have selected only subtitles and text for indexing and split the document using subtitles).



e. Run a query to check accuracy of discovery, store Discovery credentials and also the API key and URL shown step 2(a).



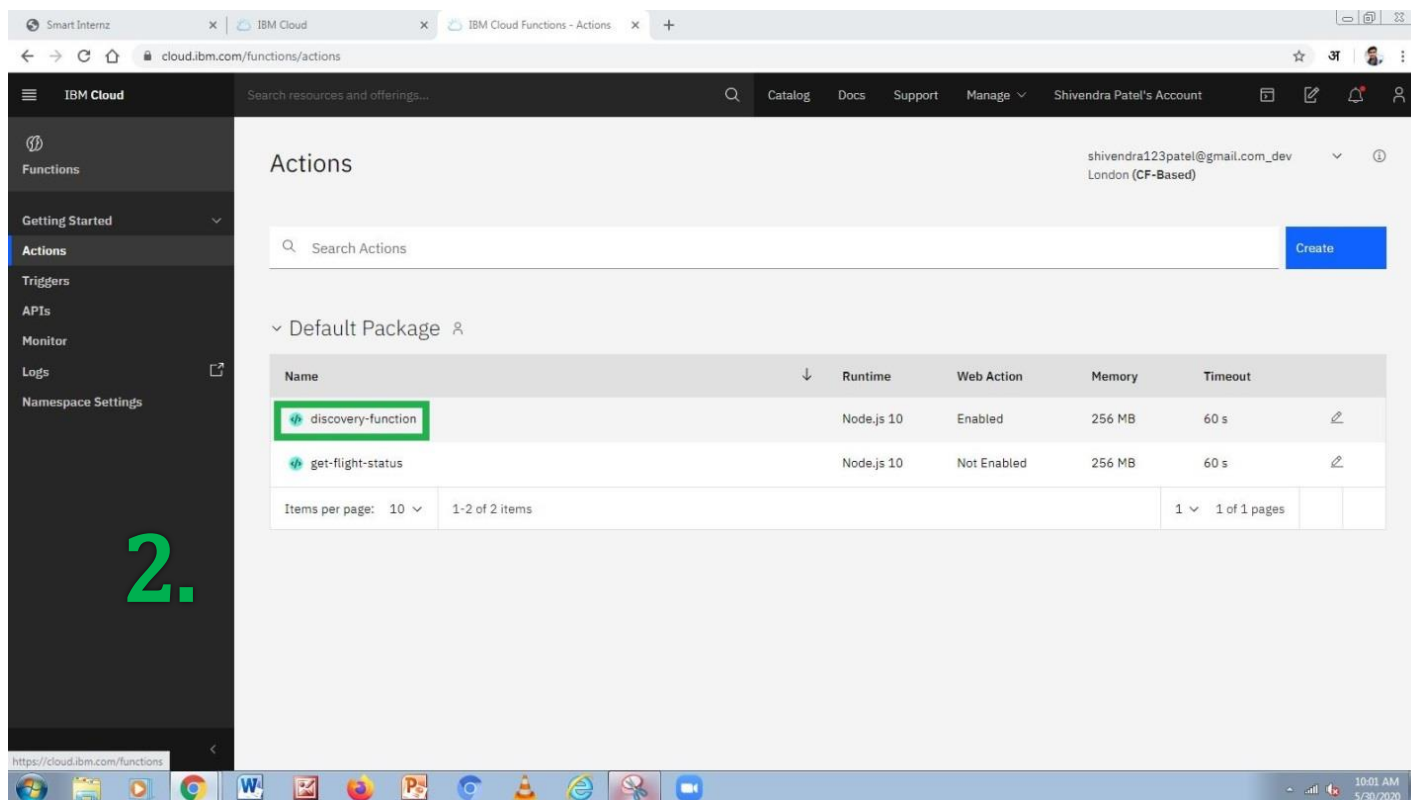
And then run your query.

The screenshot shows the IBM Watson Discovery console interface. The left sidebar contains navigation options: 'owner_manual / Build queries', 'Build a query using one or more of these components. [Learn more.](#)', 'Search for documents', 'Use natural language', 'Use the Discovery Query Language', 'how to start the heater', '+ Include analysis of your results', '+ Filter which documents you query', and '> More options'. The main area displays the 'Run query' button and a 'Close' button. The right sidebar shows the 'Summary' tab with a 'Query URL' field containing 'https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/3543'. Below this, the 'Passages' section lists several text snippets related to indoor air quality and HVAC equipment. The 'Results' section at the bottom indicates 'Showing 10 of 16 matching documents'.

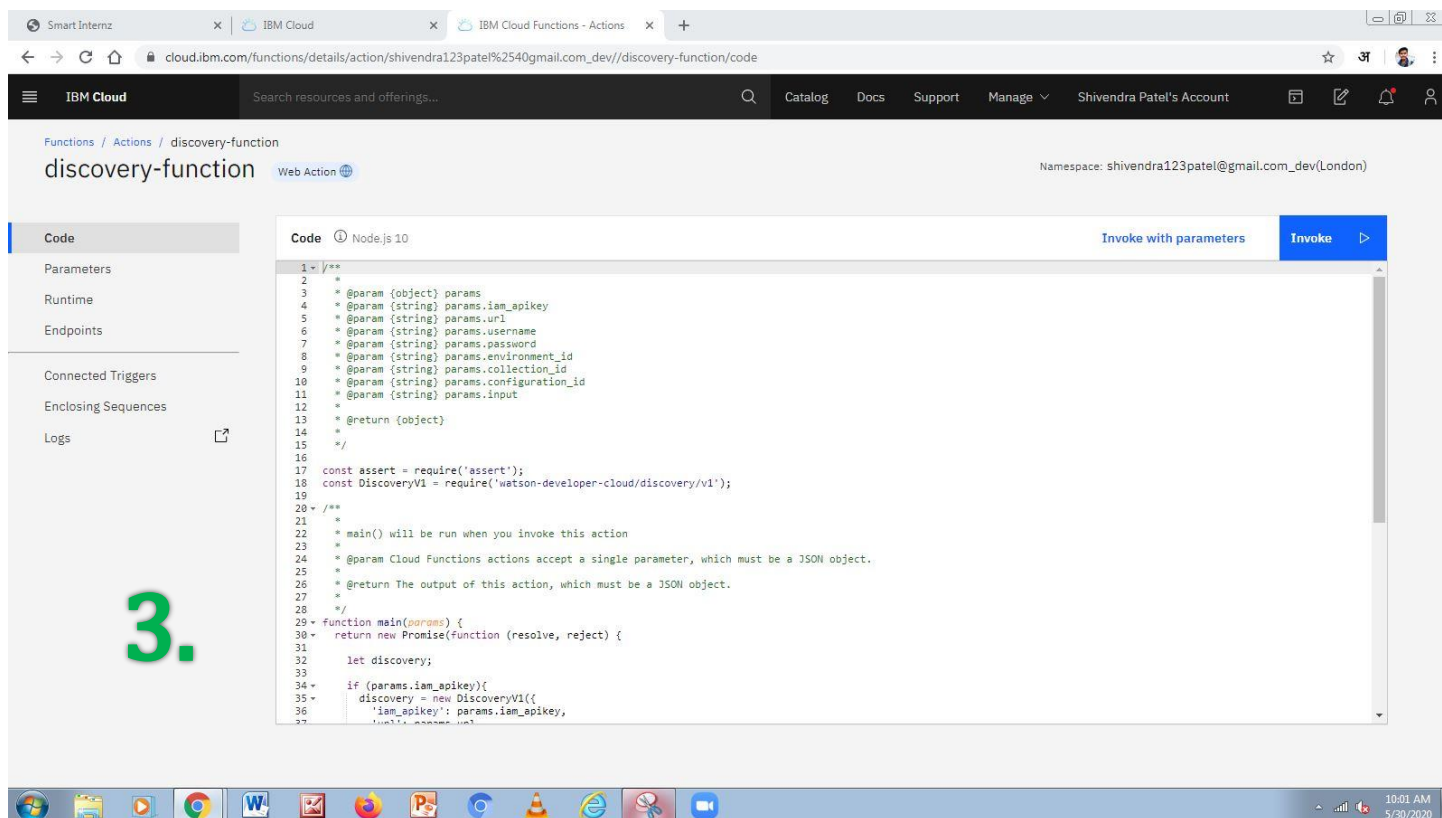
3. Create a webhook using IBM Cloud Functions Action

- a) This is used to integrate IBM Discovery Service with IBM Watson Assistant using a webhook.

The screenshot shows the IBM Cloud Functions console interface. The left sidebar contains navigation options: 'Smart Internz smartinternz.com', 'IBM Cloud', 'Functions', 'Getting Started', 'Actions' (highlighted with a green box), 'Triggers', 'APIs', 'Monitor', 'Logs', and 'Namespace Settings'. The main area displays the 'IBM Cloud Functions' header, followed by the text 'Functions-as-a-Service (FaaS) platform based on Apache OpenWhisk'. Below this, the text 'Run your application code without servers, scale it automatically, and pay nothing when it's not in use.' is shown. There are two buttons: 'Download CLI' and 'Start Creating'. The 'What's New' section lists updates: 'IAM enablement', 'Namespaces can now be explicitly managed and show up on the dashboard', 'Manage Namespace Settings', and 'View release notes'. The bottom section is titled 'Save costs, scale and integrate.' and features three icons representing different cloud services.



b. Create an IBM Cloud Functions Action (Source code is given in appendix).



c. Click on parameters, create required parameters and use the Discovery credentials for parameter values.

The screenshot shows the IBM Cloud Functions console. The left sidebar has a menu with 'Parameters' highlighted in green. The main area is titled 'discovery-function' and 'Web Action'. It displays a table of parameters with the following data:

Parameter Name	Parameter Value
url	[Redacted]
environment_id	[Redacted]
collection_id	[Redacted]
iam_apikey	[Redacted]

At the top right of the parameters section is a link 'Add Parameter'. The bottom of the image shows a Windows taskbar with various application icons and a system clock showing 10:02 AM on 5/30/2020.

d. Go to Endpoints tab and click on the Enable as web action checkbox and store the URL.

The screenshot shows the IBM Cloud Functions console with the 'Endpoints' tab selected in the left sidebar (highlighted in green). The main area shows the 'Web Action' configuration. The 'Enable as Web Action' checkbox is checked and highlighted with a green box. Below this, there is a table for HTTP endpoints:

HTTP Method	Auth	URL
ANY	Public	https://eu-gb.functions.cloud.ibm.com/[Redacted]

Below the HTTP endpoints table is a section for 'REST API' with a table:

HTTP Method	Auth	URL
POST	API-KEY	https://eu-gb.functions.cloud.ibm.com/api/v1/namespaces/shivendra123patel%40gmail.com_dev/actions/discovery-function

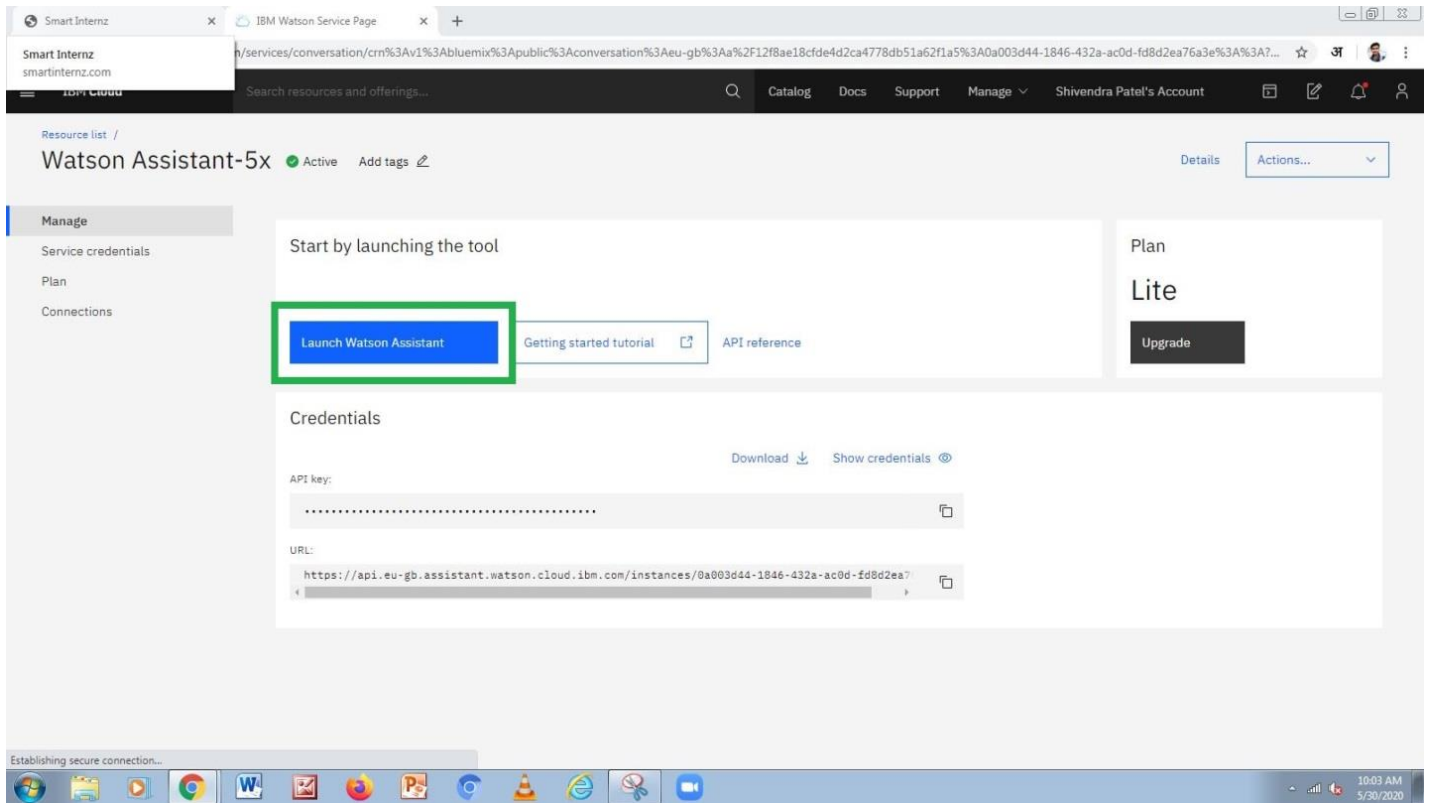
At the bottom, there is a 'CURL' section with a pre-filled curl command:

```
curl -u API-KEY -X POST https://eu-gb.functions.cloud.ibm.com/api/v1/namespaces/shivendra123patel%40gmail.com_dev/actions/discovery-function?locking=true
```

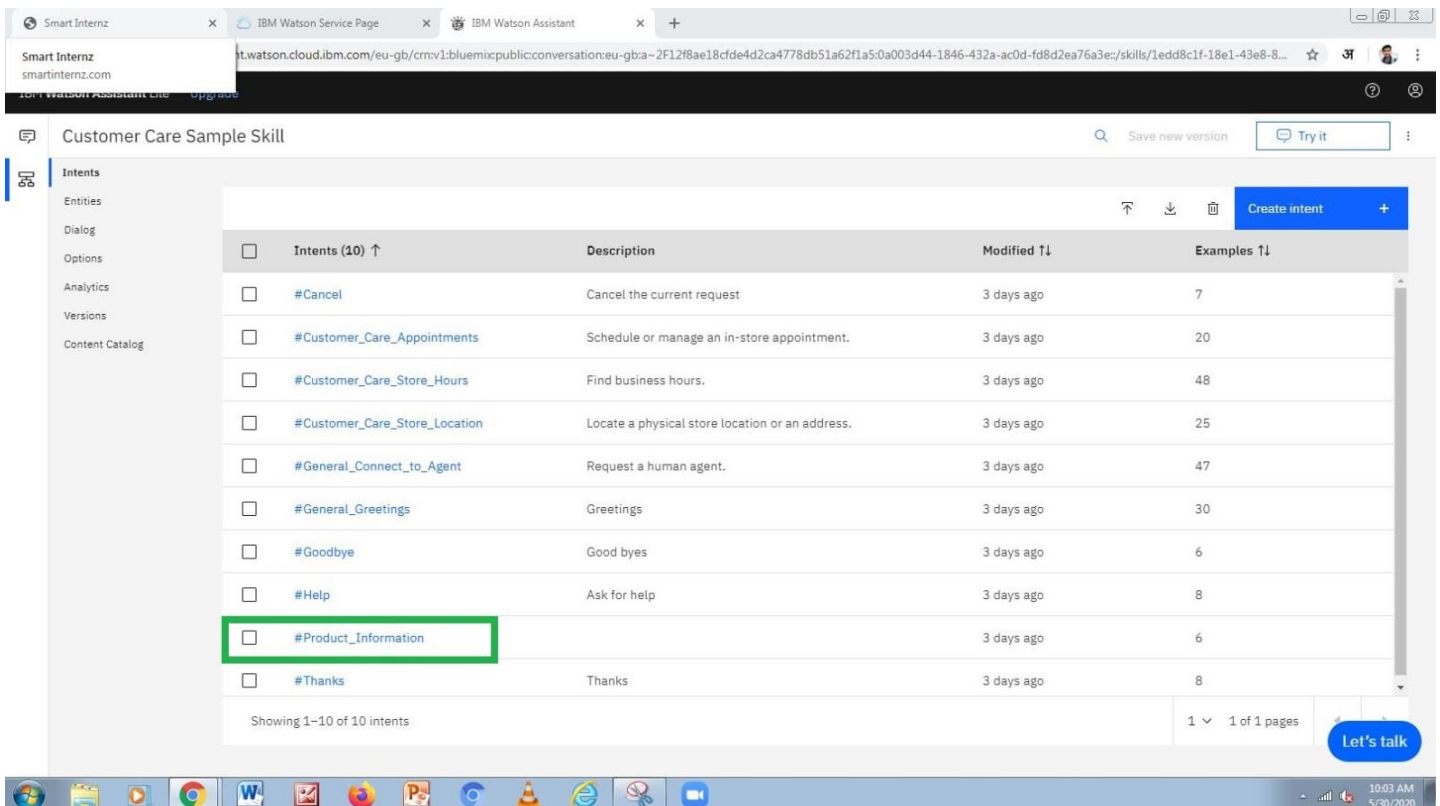
The bottom of the image shows a Windows taskbar with various application icons and a system clock showing 10:02 AM on 5/30/2020.

4. Configure IBM Watson Assistant

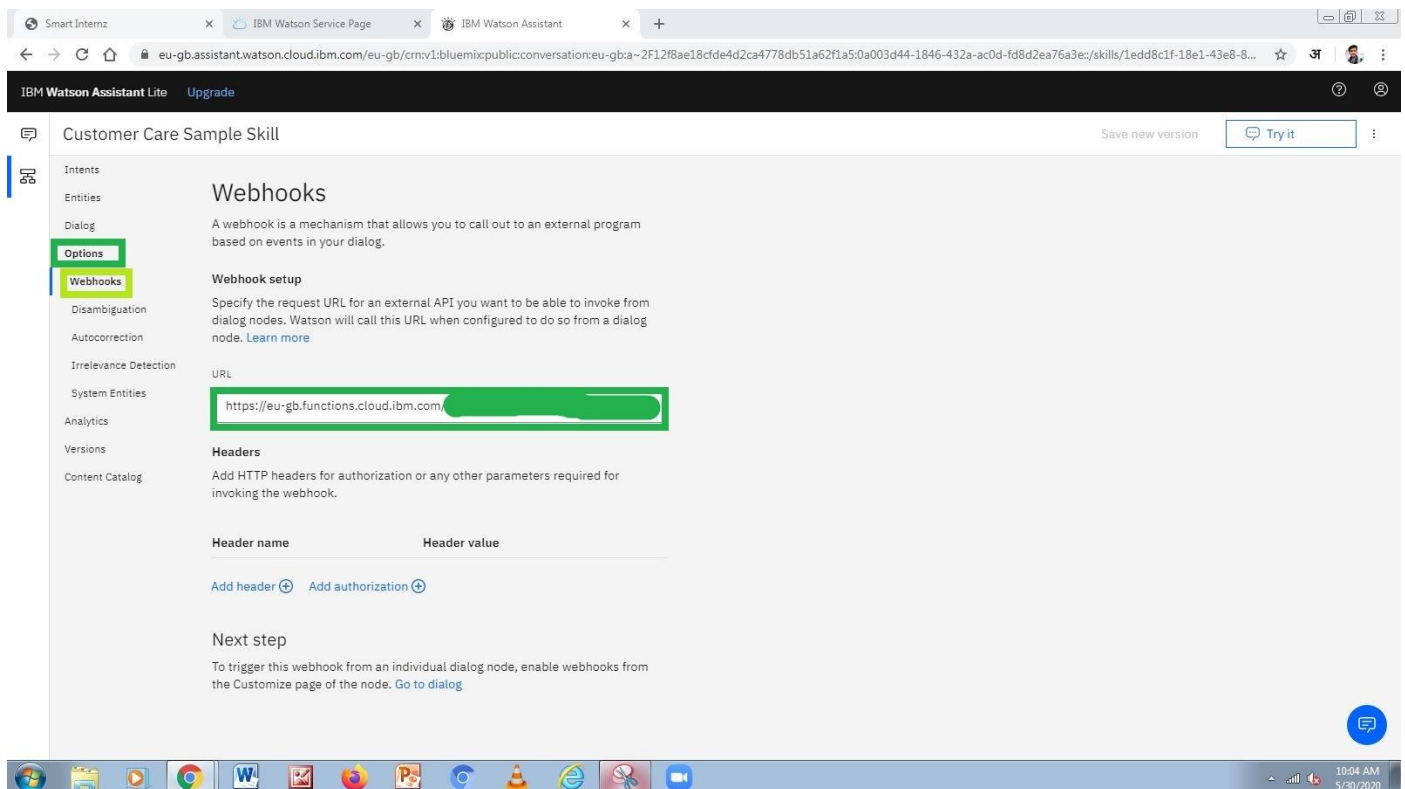
a. Launch IBM Watson Assistant and use the Customer Care Sample Skill to create an assistant with pre-loaded queries related to customer support.



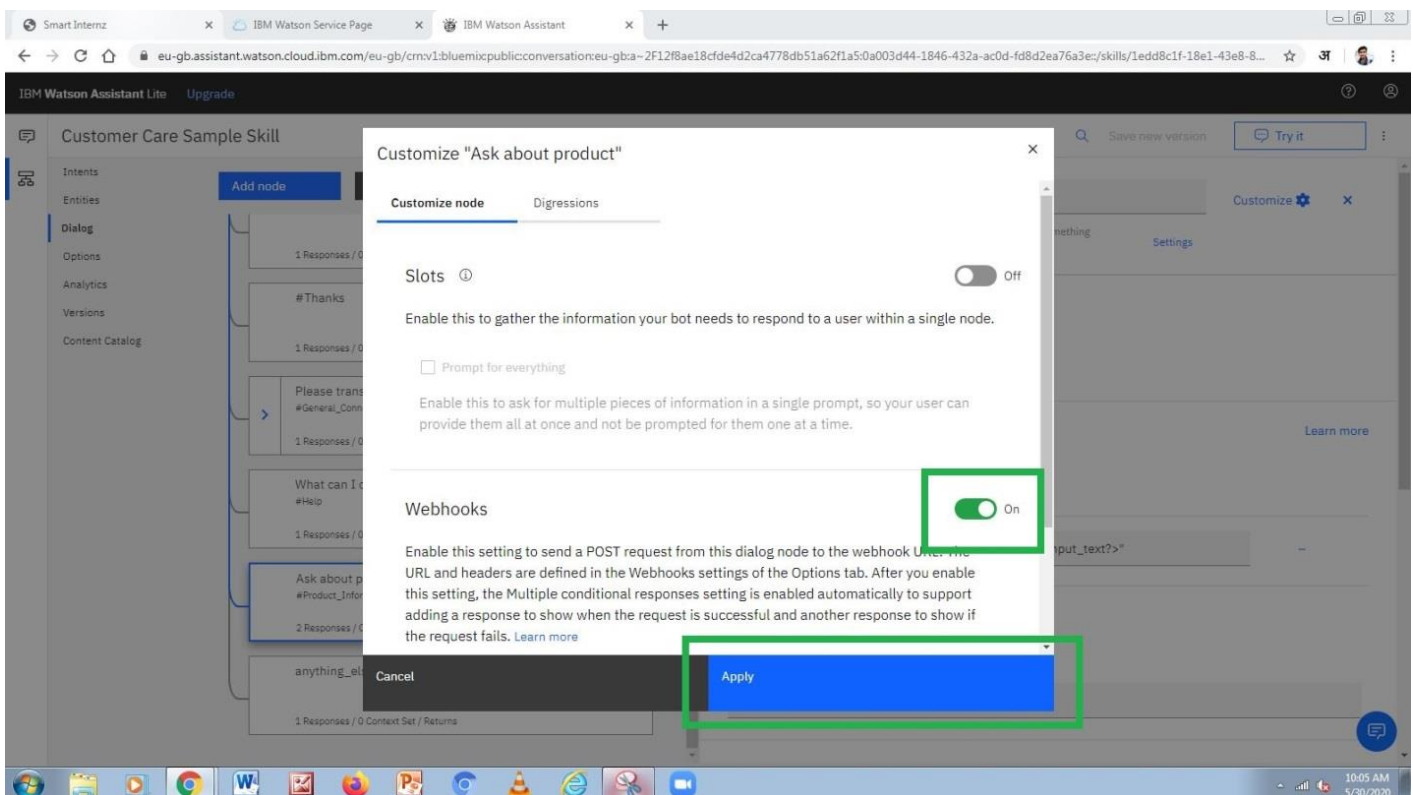
b. Create an intent for product related queries (I have created the Product Info intent) and add to dialog.



c. Go to options tab and under webhooks, paste the URL we stored in step 3(d).



d. In dialog tab enable webhooks.

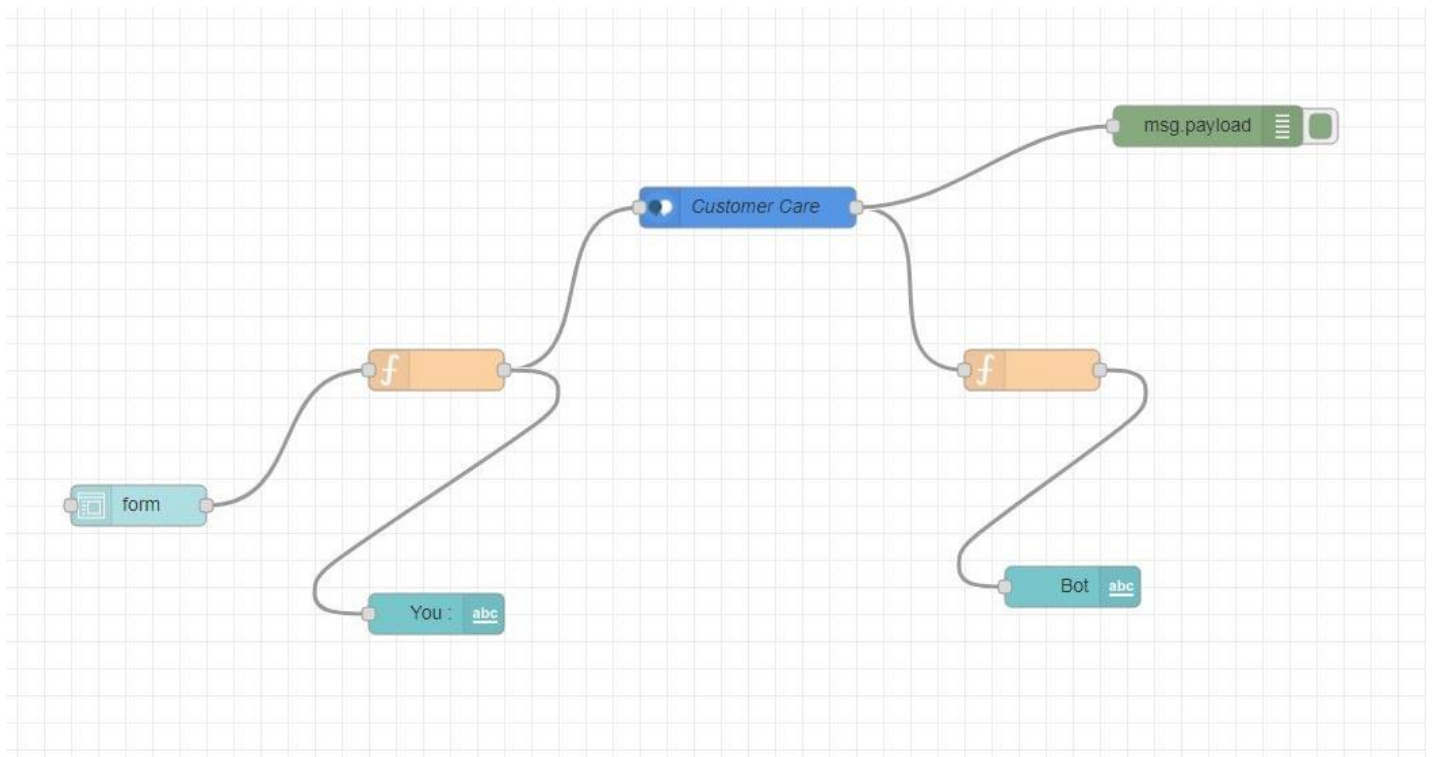


5. Create a Node-RED flow

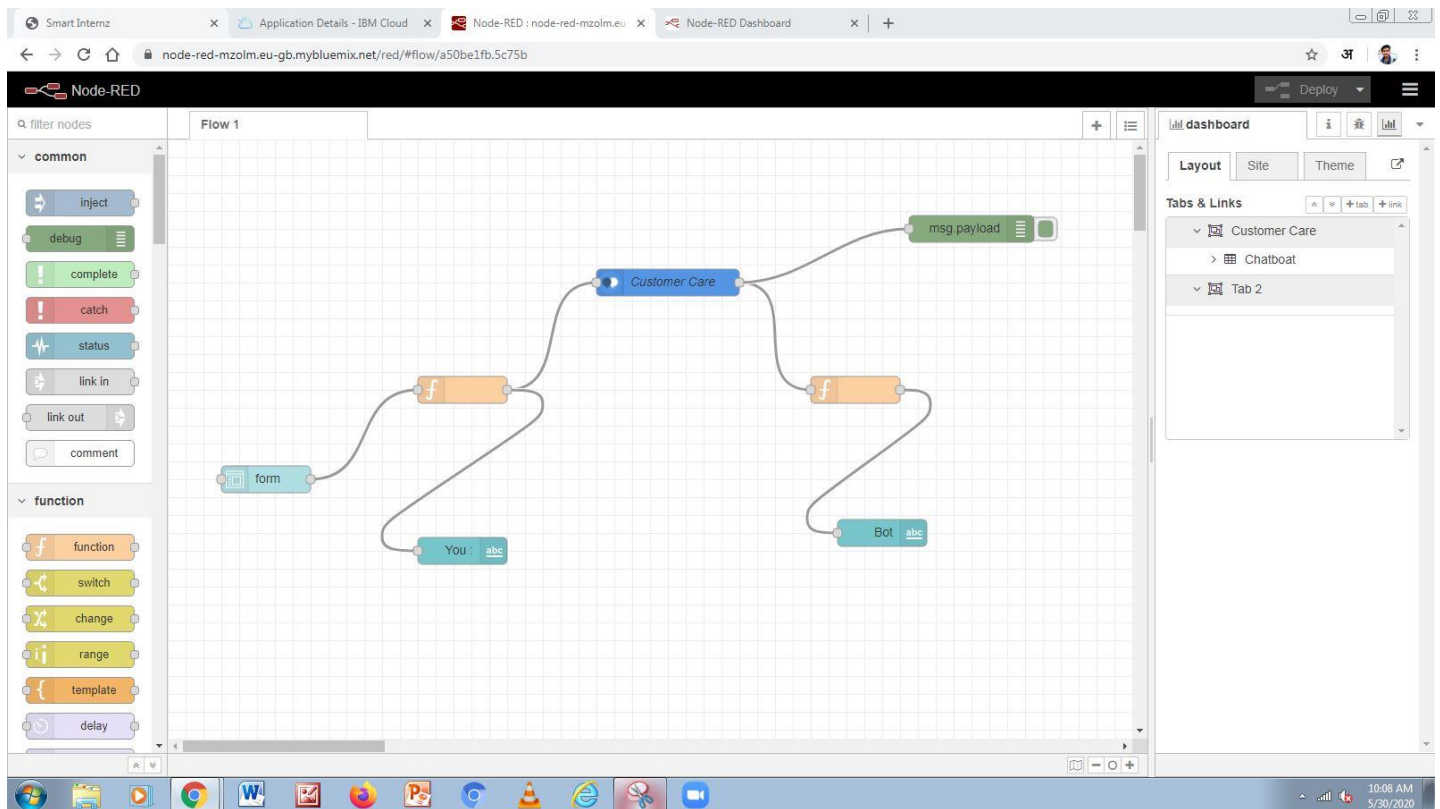
a. open Node Red app from the Cloud foundry apps.

The screenshot displays the IBM Cloud console interface for the 'Node RED MZOLM' application. The application is in a 'Running' state, indicated by a green dot and the text 'Visit App URL'. The 'Instances' section shows a health status of '100%' with '1/1 instance(s) are running'. A slider for 'MB memory per instance' is set to '256'. The 'Runtime' section shows a 'Total MB allocation' of '256' for the 'SDK for Node.js™'. The 'Runtime cost' section shows 'Current charges for billing period' as '\$0.00' and 'Estimated total for billing period' as '\$0.00'. The 'Connections (1)' section shows a single connection: 'node-red-mzolm-cloudant-1590551973084-5692'. The left sidebar contains navigation links: 'Getting started', 'Overview' (selected), 'Runtime', 'Connections', 'Logs', 'API Management', 'Autoscaling', and 'Availability Monitoring'. The top navigation bar includes 'IBM Cloud', a search bar, and links to 'Catalog', 'Docs', 'Support', 'Manage', and 'Shivendra Patel's Account'. The bottom of the image shows a Windows taskbar with various application icons and a system clock showing '10:07 AM 5/30/2020'.

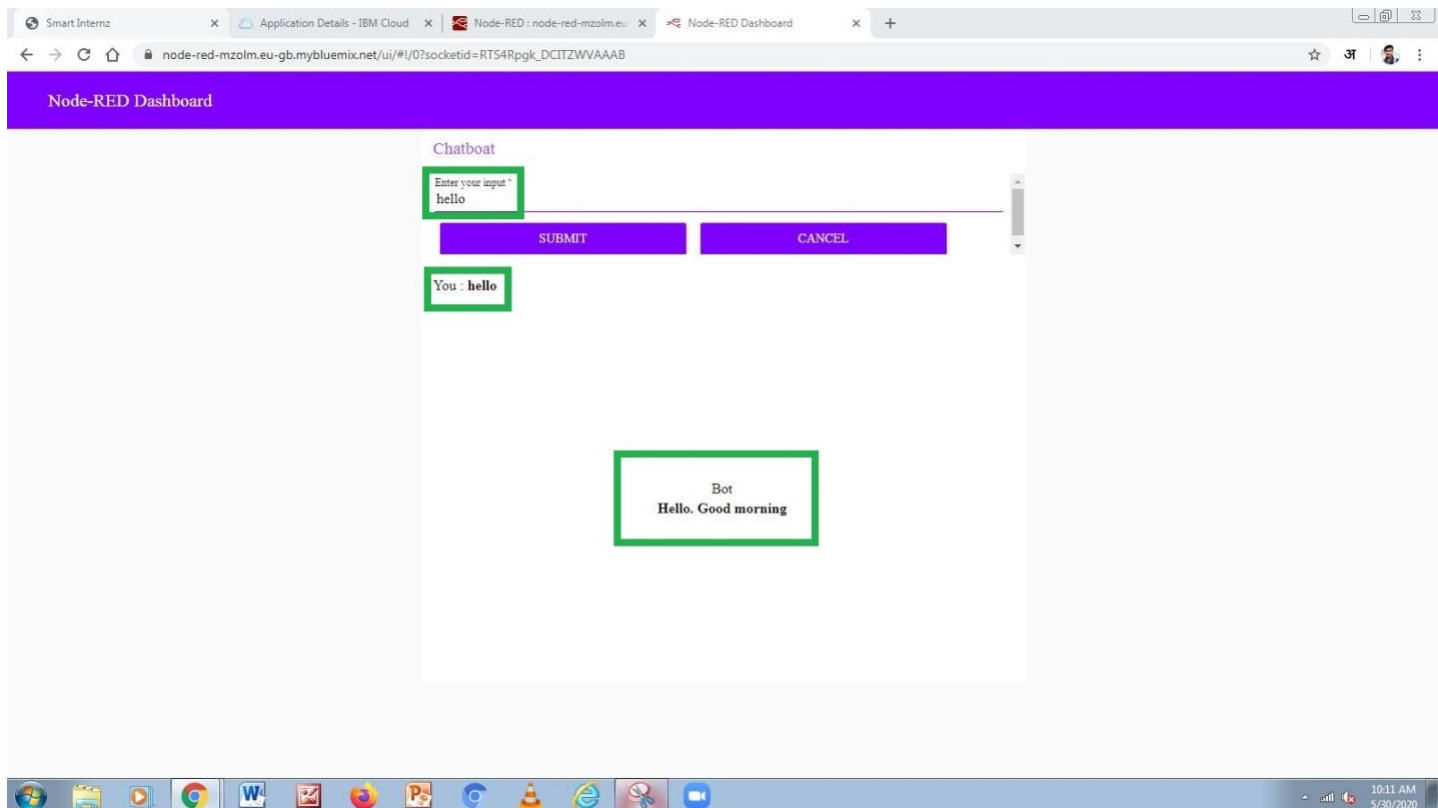
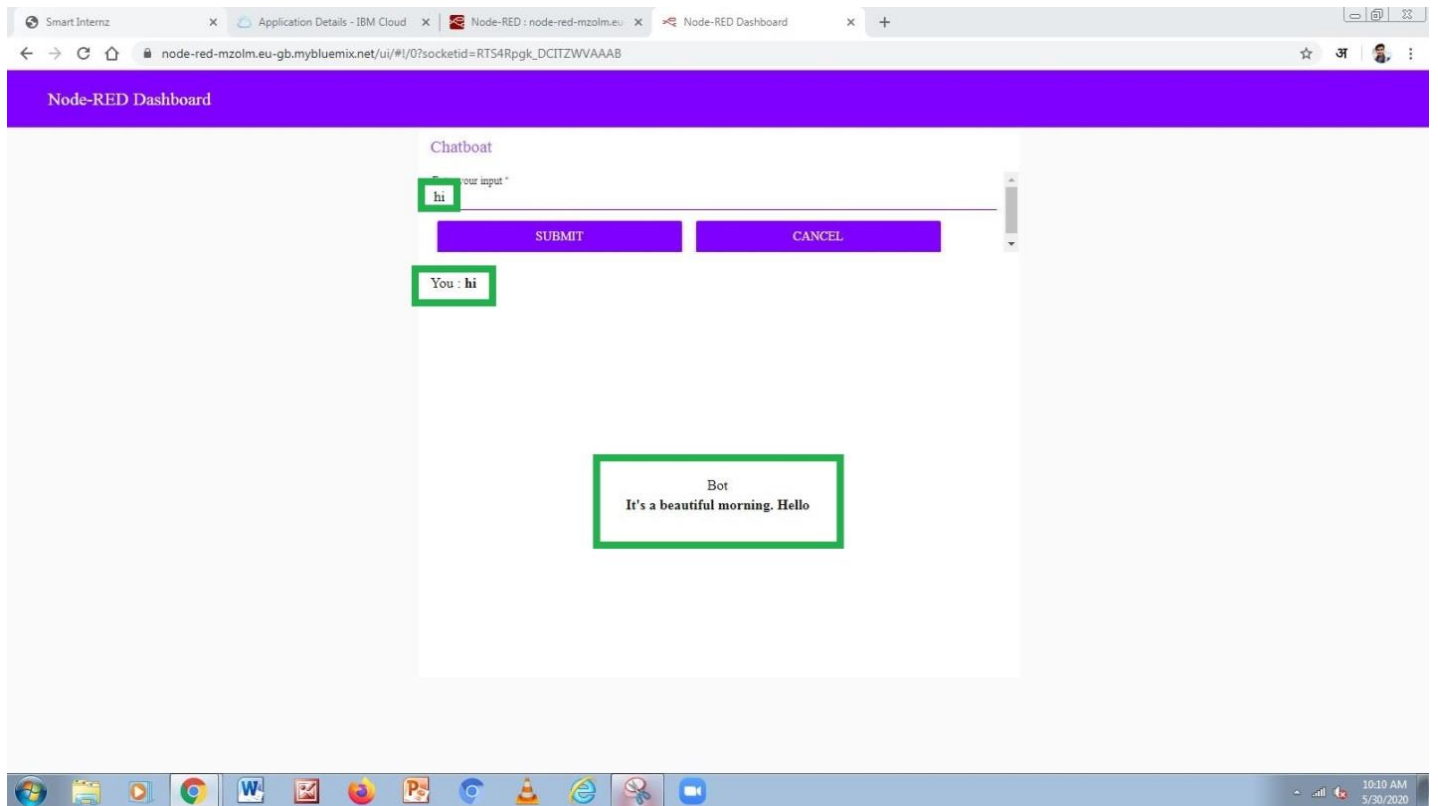
b. Use Node-RED to create a UI and to link all the services.

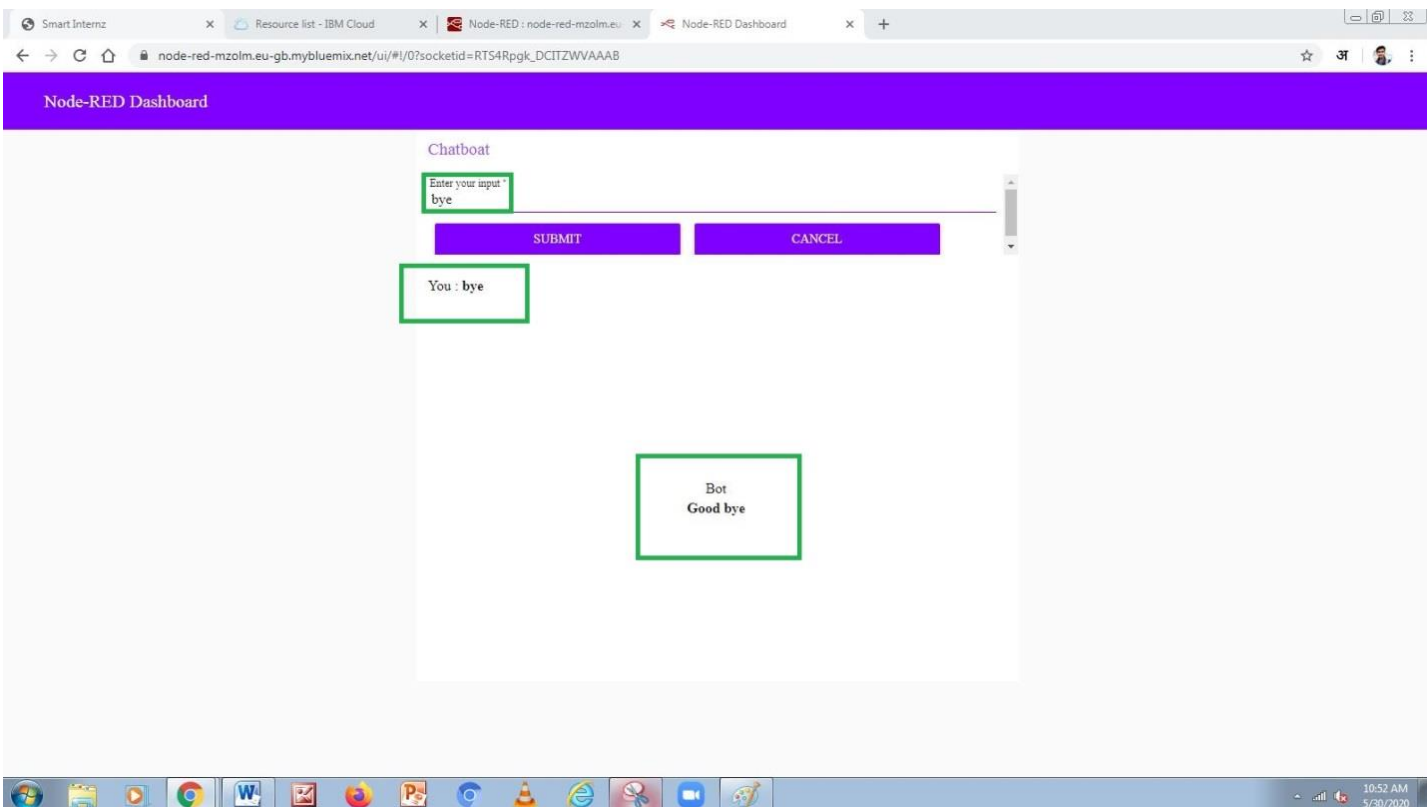
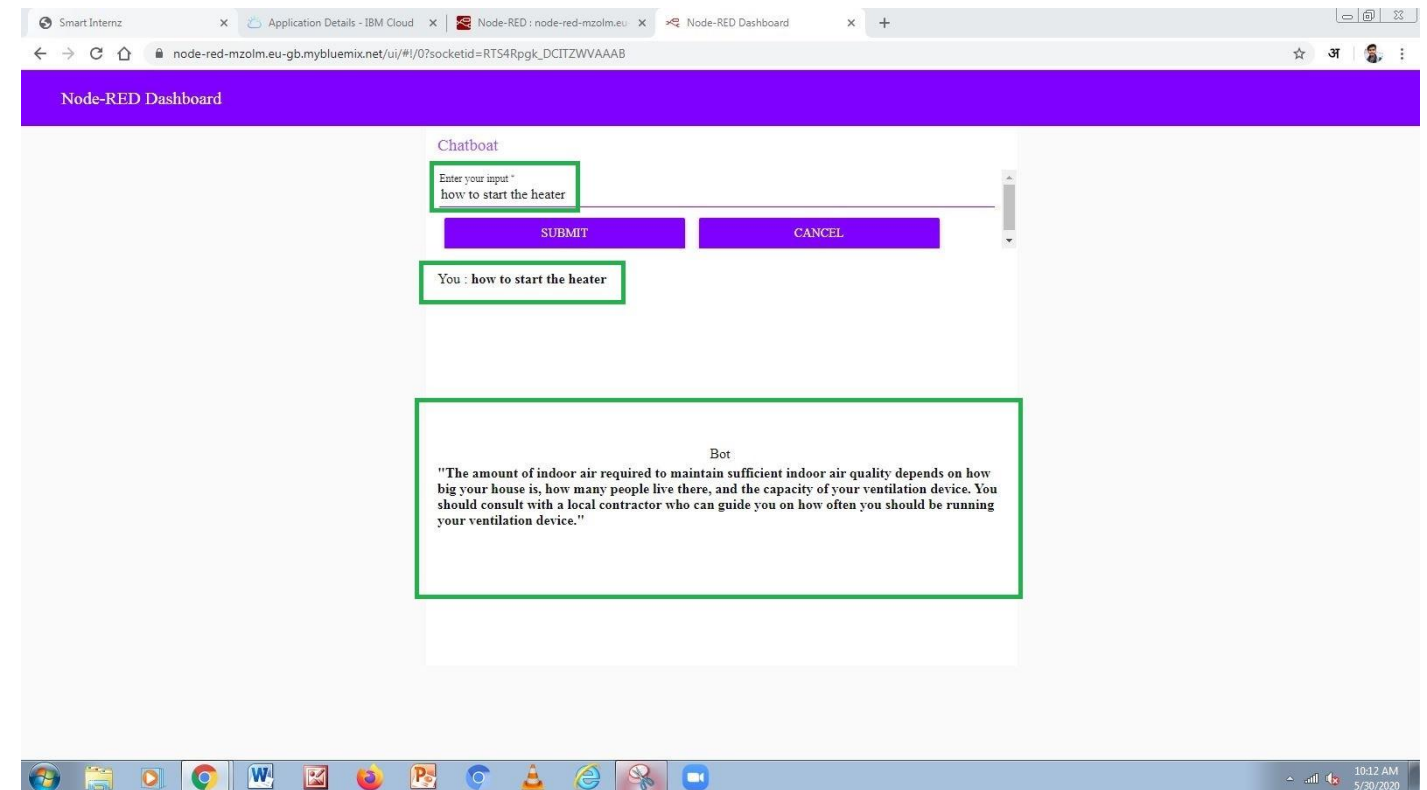


5. Flowchart



6. Result





Thus the Intelligent Customer Help Desk With Smart Document Understanding was successfully built and deployed. { <https://node-red-mzolm.eu-gb.mybluemix.net/ui> }

7. Advantages and Disadvantages

Advantages:

- Only a small amount of queries need representatives, hence the customer is also satisfied with quick and easy solution.
- Companies can use these to decrease the work flow to the representatives.
- Reduce the number of reps.
- Cost Efficient.
- Decrease in the number of calls diverted to representatives.
- Less work load on employees.

Disadvantages:

- Sometimes the chatbot misleads the customers.
- The discovery returns wrong results when not properly configured.
- Giving same answer for different sentiments.
- Sometimes is unable to connect the customer sentiments and intents.

8. Application

- It can be deployed in popular social media applications like Facebook, Slack and Telegram because of the preexisting integrations with IBM Cloud Services.
- The chatbot can be deployed in any website to clear the basic doubts of the customer.
- It can be used as a Customer Helpdesk for small scale products as their manual usually has the solution for the user's problems.

9. Conclusion

By following the above-mentioned steps, we create a basic chatbot which can help us to answer the basic questions of the customer or user related to location of the office, working hours and the information about the product. We have successfully created the intelligent help desk smart chatbot using IBM Watson Assistant, IBM Cloud Function, IBM Watson Discovery and Node-Red.

10. Future Scope

We can import the pre-built node-red flow and can improve our UI, moreover we can make a data base and use it to show the recent chats to the customer. We can also improve the results of discovery by enriching it with more fields and doing the Smart Data Annotation more accurately. We can get the premium version to increase the scope of our chatbot in terms of the calls and requests. We can also include Watson text to audio and Speech to text services to access the chatbot hands free. These are few of the future scopes which are possible.

11. Appendix

11.1. Source Code

11.1.1. IBM Cloud Functions Action (Node.js 10)

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

11.1.2 Node Red flow(.json) (formatted)

```
[
  {
    "id": "a50be1fb.5c75b",
    "type": "tab",
    "label": "Flow 1",
    "disabled": false,
    "info": ""
  },
  {
    "id": "69c2c85b.f47d48",
    "type": "ui_tab",
    "z": "",
    "name": "Customer Care",
    "icon": "dashboard",
    "disabled": false,
    "hidden": false
  },
  {
    "id": "d346fcaa.6d87f",
    "type": "ui_group",
    "z": "",
    "name": "Chatboat",
    "tab": "69c2c85b.f47d48",
    "order": 1,
    "disp": true,
    "width": 15,
    "collapse": false
  },
  {
    "id": "a6ace9b5.11d4b8",
```



```
"type": "ui_base",
"theme": {
  "name": "theme-light",
  "lightTheme": {
    "default": "#0094CE",
    "baseColor": "#8000ff",
    "baseFont": "Times New Roman,Times,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": true,
    "reset": 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",
    "reset": false
  },
  "themeState": {
    "base-color": {
      "default": "#0094CE",
      "value": "#8000ff",
      "edited": true
```

```
},
"page-titlebar-backgroundColor": {
  "value": "#8000ff",
  "edited": false
},
"page-backgroundColor": {
  "value": "#fafafa",
  "edited": false
},
"page-sidebar-backgroundColor": {
  "value": "#000000",
  "edited": false
},
"group-textColor": {
  "value": "#a64dff",
  "edited": false
},
"group-borderColor": {
  "value": "#ffffff",
  "edited": false
},
"group-backgroundColor": {
  "value": "#ffffff",
  "edited": false
},
"widget-textColor": {
  "value": "#111111",
  "edited": false
},
"widget-backgroundColor": {
  "value": "#8000ff",
```

```
    "edited": false
  },
  "widget-borderColor": {
    "value": "#ffffff",
    "edited": false
  },
  "base-font": {
    "value": "Times New Roman,Times,serif"
  }
},
"angularTheme": {
  "primary": "indigo",
  "accents": "blue",
  "warn": "red",
  "background": "grey"
}
},
"site": {
  "name": "Node-RED Dashboard",
  "hideToolbar": "false",
  "allowSwipe": "false",
  "lockMenu": "true",
  "allowTempTheme": "true",
  "dateFormat": "DD/MM/YYYY",
  "sizes": {
    "sx": 30,
    "sy": 37,
    "gx": 0,
    "gy": 0,
    "cx": 16,
    "cy": 16,
```

```
        "px": 0,
        "py": 0
    }
}
},
{
    "id": "3530ddaf.f6d482",
    "type": "ui_tab",
    "name": "Tab 2",
    "icon": "dashboard",
    "order": 2
},
{
    "id": "6992a746.8ff768",
    "type": "watson-conversation-v1",
    "z": "a50be1fb.5c75b",
    "name": "Customer Care",
    "workspaceid": "1edd8c1f-18e1-43e8-8dc3-709bc13a8914",
    "multiuser": false,
    "context": true,
    "empty-payload": false,
    "service-endpoint": "https://api.eu-
gb.assistant.watson.cloud.ibm.com/instances/0a003d44-1846-432a-ac0d-
fd8d2ea76a3e",
    "timeout": "",
    "optout-learning": false,
    "x": 560,
    "y": 160,
    "wires": [
        [
            "403d959b.941acc",
```

```
        "a36765c3.e52628"
    ]
]
},
{
    "id": "36945a85.7422f6",
    "type": "ui_text",
    "z": "a50be1fb.5c75b",
    "group": "d346fcaa.6d87f",
    "order": 3,
    "width": 15,
    "height": 8,
    "name": "",
    "label": "Bot",
    "format": "{{msg.payload}}",
    "layout": "col-center",
    "x": 800,
    "y": 440,
    "wires": []
},
{
    "id": "727b545b.8890cc",
    "type": "ui_text",
    "z": "a50be1fb.5c75b",
    "group": "d346fcaa.6d87f",
    "order": 2,
    "width": 0,
    "height": 0,
    "name": "",
    "label": "You :",
    "format": "{{msg.payload}}",
```

```
"layout": "row-left",
"x": 330,
"y": 460,
"wires": []
},
{
  "id": "a3dac9fe.b52f38",
  "type": "function",
  "z": "a50be1fb.5c75b",
  "name": "",
  "func": "msg.payload=msg.payload.text;\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "x": 330,
  "y": 280,
  "wires": [
    [
      "6992a746.8ff768",
      "727b545b.8890cc"
    ]
  ]
},
{
  "id": "403d959b.941acc",
  "type": "function",
  "z": "a50be1fb.5c75b",
  "name": "",
  "func": "msg.payload=msg.payload.output.text[0];\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "x": 770,
```

```
"y": 280,
"wires": [
  [
    "36945a85.7422f6"
  ]
],
{
  "id": "a36765c3.e52628",
  "type": "debug",
  "z": "a50be1fb.5c75b",
  "name": "",
  "active": true,
  "tosidebar": true,
  "console": false,
  "tostatus": false,
  "complete": "false",
  "x": 900,
  "y": 100,
  "wires": []
},
{
  "id": "29abb804.e31a48",
  "type": "ui_form",
  "z": "a50be1fb.5c75b",
  "name": "",
  "label": "",
  "group": "d346fcaa.6d87f",
  "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": 110,  
  "y": 380,  
  "wires": [  
    [  
      "a3dac9fe.b52f38"  
    ]  
  ]  
}  
]
```


11.2 References

- https://www.ibm.com/cloud/architecture/tutorials/cognitive_discovery
- <https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started>
- <https://developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chatbot-on-nodered/>
- <http://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red>
- <https://github.com/IBM/watson-discovery-sdu-with-assistant>