A

Project Report

On

Intelligent Customer Help Desk

with

Smart Document Understanding

Internship under:

SMARTBRIDGE

NAME : TUSHAR ANAND

EMAIL-ID: ta2091994@gmail.com

PROJECT ID : SPS_PRO_99

PROJECT TITLE : Intelligent Customer Help Desk with Smart Document Understanding

Category: Artificial Intelligence

Table Of Content

DESCRIPTION	PAGE NO.
1. INTRODUCTION	
Overview	3
Purpose	4
2. LITERATURE SURVEY	
Existing Problem	5
Proposed solution	5
3. THEORITICAL ANALYSIS	
Block Diagram	6
Hardware/Software design	7
4. EXPERIMENTAL INVESTIGATION	8
SCREENSHOTS	18
5. FLOWCHART	22
6. RESULT	23
7. ADVANTAGES & DISADVANTAGES	25
8. APPLICATION	26
9. CONCLUSION	26
10. FUTURE SCOPE	26
8. BIBILIOGRAPHY	27
APPENDIX	
A. SOURCE CODE	

1. INTRODUCTION

Overview:

This project “Intelligent Customer Help Desk With Smart Document Understanding” is a an web application that combines multiple Watson AI Services (Discovery , Assistant, Cloud function and Node Red).

In this project, we learn best practices of combining Watson services, and how they can build interactive information retrieval systems with Discovery + Assistant.

- Project Requirements: IBM Cloud, IBM Watson, IBM Discovery, Node- RED
- Functional Requirements: IBM cloud
- Technical Requirements: WATSON AI
- Software Requirements: Python, Watson assistant, Watson discovery.
- Project Deliverables: Smartinternz Intership
- Project Team: Ankit Sahu
- Project Duration:28 days

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, 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 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.

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

Existing problem:

For businesses, it has become necessary to solve the queries and problems of the customers to ensure consumer loyalty along with the brand establishment.

In our regular customer support team, there are many problems exist, such as :

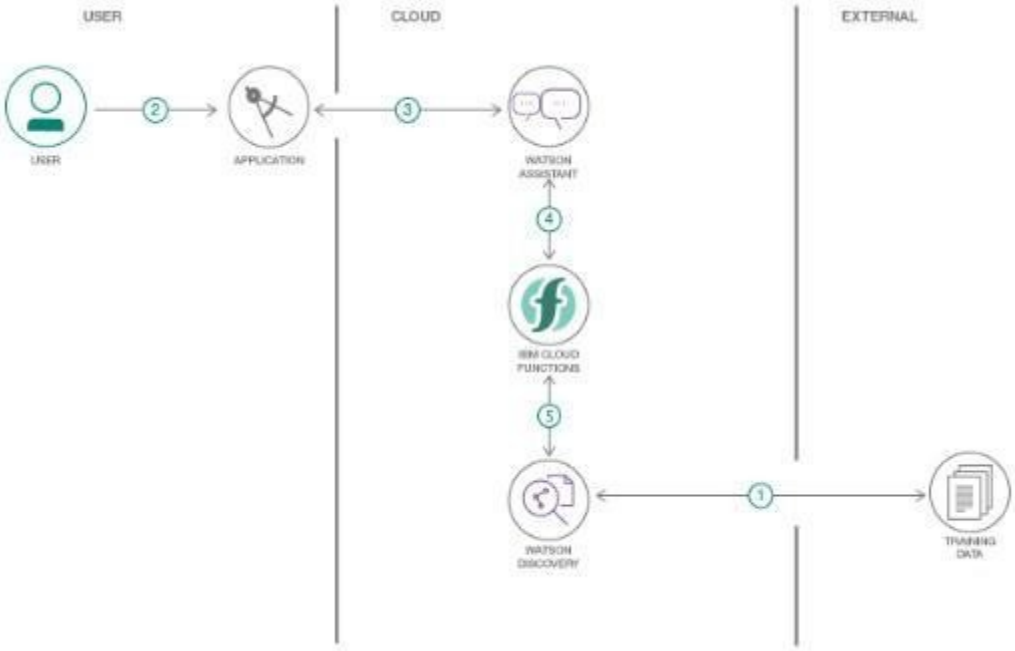
- Generally live agents can handle only 2 to 3 conversations at a time,
- If our business receives a lot of inquires(may be all queries are similar or repeated), we also need high customer support team.
- 365X24X7 Support — our support staff need any breaks, they can't work 24 hours a day.

Proposed solution:

For the above problem to get solved we have to put an virtual agent in CHATBOT so it can understand the queries that are posted by customers. The virtual agent should trained from some insight records based company background so it can answer queries based on the product or related to company. In this project I used Watson Discovery to achieve the above solution. And later including Assistant and Discovery on Node-RED.

3. THEORITICAL ANALYSIS

Block/Flow Diagram:



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.

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 Assistant
- Watson Discovery
- IBM cloud function
- Node Red

i) CREATING WATSON ASSISTANT

Before we begin , we needs a service instance to start :

a. Go to the Watson Assistant page in the IBM Cloud™ catalog.
b. Sign up for a free IBM Cloud account or log in.
c. Click **Create**.

STEP 1 : OPEN WATSON ASSISTANT

After we create a Watson Assistant service instance, we land on the Manage page of the Watson Assistant dashboard.

 Click **Launch Watson Assistant**. If you're prompted to log in, provide your IBM Cloud credentials.

A new browser tab or window opens and the Assistants page of Watson Assistant is displayed.

- An assistant named My first assistant is created for you automatically. An assistant is a cognitive bot to which you add skills that enable it to interact with your customers in useful ways.

- A dialog skill named My first skill is added to the assistant for you automatically. A dialog skill is a container for the artifacts that define the flow of a conversation that your assistant can have with your customers.

- If an assistant and skill are not created automatically, complete Steps 2 and 3. Otherwise, skip to Step 4: Add intents from a content catalog.

STEP 2 : CREATE AN ASSISTANT

An assistant is a cognitive bot to which you add skills that enable it to interact with your customers in useful ways.

- Click the Assistants icon Assistants menu icon, and then click Create assistant.
- Create assistant button on the Assistants page.
- Name the assistant My first assistant.
- Finish creating the new assistant
- Click Create assistant.

STEP 3 : CREATE A DIALOG SKILL

A dialog skill is a container for the artifacts that define the flow of a conversation that your assistant can have with your customers.

- Click the My first assistant tile to open the assistant.
- Click Add dialog skill.
- Shows the Add skill button from the home page
- Give your skill the name My first skill.
- Optional. If the dialog you plan to build will use a language other than English, then choose the appropriate language from the list.
- Finish creating the skill
- Click Create dialog skill.
- The skill is created and you return to the assistant page.
- Finish creating the skill
- Click to open the skill you just created.

STEP 4 : ADD INTENTS FROM A CONTENT CATALOG

When you open the My first skill, you land on the Intents

page. Shows the Intents page of My first skill

If available in your location, a tour begins that you can step through to learn about the product. Follow the tour; it provides a great overview of the product.

Add training data that was built by IBM to your skill by adding intents from a content catalog. In particular, you will give your assistant access to the General content catalog so your dialog can greet users, and end conversations with them.

- Click the Content Catalog tab.
- Find General in the list, and then click Add to skill.
- Shows the Content Catalog and highlights the Add to skill button for the General catalog.
- Open the Intents tab to review the intents and associated example utterances that were added to your training data. You can recognize them because each intent name begins with the prefix #General_. You will add the #General_Greetings and #General_Ending intents to your dialog in the next step.
- Shows the intents that are displayed in the Intents tab after the General catalog is added.

- You successfully started to build your training data by adding prebuilt content from IBM.

STEP 5 : BUILD A DIALOG

A dialog defines the flow of your conversation in the form of a logic tree. It matches intents (what users say) to responses (what the bot says back). Each node of the tree has a condition that triggers it, based on user input.

We'll create a simple dialog that handles greeting and ending intents, each with a single node.

Adding a start node

- From the Skills menu, click Dialog.
- Click Create dialog. You see two nodes:
    - Welcome: Contains a greeting that is displayed to your users when they first engage with the assistant.
    - Anything else: Contains phrases that are used to reply to users when their input is not recognized.
- Click the Welcome node to open it in the edit view.
- Replace the default response with the text, Welcome to the Watson Assistant tutorial!.
- Click Close to close the edit view.

We created a dialog node that is triggered by the welcome condition. (welcome is a special condition that functions like an intent, but does not begin with a #.) It is triggered when a new conversation starts. Our node specifies that when a new conversation starts, the system should respond with the welcome message that we add to the response section of this first node.

STEP 6 : INTEGRATE THE ASSISTANT

Now that you have an assistant that can participate in a simple conversational exchange, test it.

- Click the Assistants icon Assistants menu icon to open a list of your assistants.
- Find the My first assistant assistant, and open it.
- Test your assistant with a preview link integration.

The preview link integration builds your assistant into a chat widget that is hosted by an IBM- branded web page. You can open the web page and chat with your assistant to test it out.

- If a My first assistant assistant was created for you, then you must add a preview link integration. From the Integrations area, click Add integration, and then click Preview Link. Click Create.

When you you create an assistant yourself, you can skip this step. A preview link integration is added to the assistant for you automatically. Click the preview link integration tile to open it.

- Click the URL that is displayed on the page.

ii) CREATING WATSON DISCOVERY

Before we begin, we need a service instance to

start. Go to the Discovery page in the IBM Cloud

catalog.

The service instance is created in the default resource group if you do not choose a different one, and it cannot be changed later. This group is sufficient for the purposes of trying out the service.

- If you're creating an instance for more robust use, then learn more about resource groups.
- Sign up for a free IBM Cloud account or log in.
- Click Create.

STEP 1 : LAUNCH THE TOOLING

After you create an instance of Discovery, you're taken to your list of services.

- Click the Discovery instance you created to go to the service dashboard.
- On the Manage page, click Launch Watson Discovery. If you're prompted to log in to the tooling, provide your IBM Cloud credentials.

STEP 2 : CREATE A COLLECTION

- Your first step in the Discovery tooling is to create a data collection.
- A collection is a set of your documents. Why would I want more than one collection? There are a few reasons:
    - You might want multiple collections to separate results for different audiences.
    - The data might be so different that it doesn't make sense for it all to be queried at the same time.

The public, pre-enriched Discovery News data collection is also available for your use. It is ready to query, and you can begin to create queries on it immediately. You cannot adjust its configuration or add documents to Discovery News.

- Click Environment details and choose Create environment.
- When your environment is ready, click the Upload your own data button, then you can Name your new collection. Name your collection Install Docs.

STEP 3 : DOWNLOAD THE SAMPLE DOCUMENT AND UPLOAD TO YOUR COLLECTION

- Download this sample PDF document: Watson Explorer Installation Guide External link icon. See Supported document types for the full list of document types supported in Discovery.

  - Upload the document to your collection. Either drag and drop it into your collection, or click **browse from computer** to upload documents. After the upload is complete, the following information displays:
    - The number of documents (1).
    - The fields identified from your document. You should see one field identified, text. We identify additional fields in a bit.
    - Enrichments applied to your document. The Entity Extraction, Sentiment Analysis, Category Classification, and Concept Tagging enrichments are automatically applied to the text field by Discovery. For more information about enrichments, see Adding enrichments.
    - Pre-built queries you can run immediately.
  - Let's try a quick Natural Language Query to level set. Click **Build your own query** on the lower right.
  - On the **Build queries** screen, click on **Search for documents**, then **Use natural language**. Enter What are the minimum hardware requirements and click the **Run query** button. Click the **JSON** tab on the right. The result is not as precise as it could be, so let's improve it with Smart Document Understanding.
  - Click on the name of the collection (**InstallDocs**) on the upper left to return to the **Overview** screen.

STEP 4 : Annotate your document

- Click Configure data on the upper right.
- On the Configure data screen, there are three tabs: Identify fields, Manage fields, and Enrich fields.
- The Watson Explorer Installation Guide is displayed and ready for annotation on the Identify fields tab. All available fields (answer, author, footer, header, question, subtitle, table_of_contents, text, and title) are displayed in the Field labels list on the right. If you purchase an Advanced or Premium plan you can create your own custom labels.
- Click on title, then select the marker next to Installation and Integration Guide. Click **Submit page**.
- In the page preview on the left, click on page 3. Note that the title is already predicted for this page. Click **Submit page**.
- On page 4, select the footer label and select the marker next to the footer. Click the **Submit page** button.
- On pages 5 and 6, annotate the footers with the footer label. Submit each page. Click through a few more pages; note that the footer was predicted properly by Discovery. Annotate the titles (flush left) and subtitles (indented) on pages 7, 9, and 10 and submit each page individually.
- Click through a few more pages and check the predicted titles and subtitles. If any need to be changed, annotate those pages, and click **Submit page**.
- Now click on the **Manage fields** tab and under **Improve query results by splitting your documents** split the document, based on subtitle.
- Click the **Apply changes to collection** button on the top right. An **Upload your documents** dialog box appears. Browse to the original watsonexplorerinstall.pdf file, and upload it. All of the annotations are applied to your index. After it finishes indexing, the **Overview** screen opens. Expect to now see more than 30 documents and 4 fields identified from your data: footer, subtitle, text, and title. If the changes do not display within a few minutes, refresh the browser window.

STEP 5 : LET'S QUERY:

- Click Build your own query on the bottom right.
- On the Build queries screen, click on Search for documents, then Use natural language. Enter What are the minimum hardware requirements and click the Run query button.
- Click the JSON tab on the right. Look at the text under results. The answers returned for the query are much more precise.

iii) CREATING NODE RED APP

STEP 1 : FIND THE NODE-RED STARTER IN THE IBM CLOUD CATALOG

Follow these steps to create a Node-RED Starter application in the IBM Cloud.

- Log in to IBM Cloud.
- Open the catalog and search for node-red .
- Click on the Software tab .
- Click on the Node-RED App tile .
- Catalog entry Node-RED Starter Kit
- Click on the Create app button to continue.

STEP 2 : CONFIGURE YOUR APPLICATION

Now you need to configure the Node-RED Starter application.

- On the App details page, a randomly generated name will be suggested – Node RED SSLPD in the screenshot below. Either accept that default name or provide a unique name for your application . This will become part of the application URL. Note: If the name is not unique, you will see an error message and you must enter a different name before you can continue.
- The Node-RED Starter application requires an instance of the Cloudant database service to store your application flow configuration. Select the region the service should be created in and what pricing plan it should use. Note: You can only have one Cloudant instance using the Lite plan. If you have already got an instance, you will be able to select it from the Pricing plan select box . You can have more than one Node-RED Starter application using the same Cloudant service instance.

- Click the Create button to continue. This will create your application, but it is not yet deployed to IBM Cloud.

STEP 3 : ENABLE THE CONTINUOUS DELIVERY FEATURE

- At this point, you have created the application and the resources it requires, but you have not deployed it anywhere to run. This step shows how to setup the Continuous Delivery feature that will deploy your application into the Cloud Foundry space of IBM Cloud.
- On the next screen, click the Deploy your app button to enable the Continuous Delivery feature for your application.
- Enable continuous delivery in Node-RED app
- You will need to create an IBM Cloud API key to allow the deployment process to access your resources. Click the New button to create the key. A message dialog will appear. Read what it says and then confirm and close the dialog.
- Increase the Memory allocation per instance slider to 256MB. If you do not increase the memory allocation, your Node-RED application might not have sufficient memory to run successfully.
- The Node-RED Starter kit only supports deployment to the Cloud Foundry space of IBM Cloud. Select the region to deploy your application to. This should match the region you created your Cloudant instance in. Lite users might only be able to deploy to your default region.
- Select the region to create the DevOps toolchain.
- Click Create. This will take you back to the application details page.
- Create the Node-RED Starter app
- After a few moments, the Continuous Delivery section will refresh with the details of your newly created Toolchain. The Status field of the Delivery Pipeline will show In progress. That means your application is still being built and deployed.
- Continuous delivery status
- Click on the In progress link to see the full status of the Delivery Pipeline.
- Delivery pipeline, view logs
- The Deploy stage will take a few minutes to complete. You can click on the View logs and history link to check its progress. Eventually the Deploy stage will go green to show it has passed. This means your Node-RED Starter application is now running.

STEP 4: OPEN THE NODE-RED APPLICATION

Now that you've deployed your Node-RED application, let's open it up!

- Open your IBM Cloud Resource list by selecting the sidebar menu (1) and then selecting Resource List .
- You will see your newly created Node-RED Application listed under the Apps section (1). You will also see a corresponding entry under the Cloud Foundry apps section (2). Click on this Cloud Foundry app entry to go to your deployed application's details page.

- From the details page, click the Visit App URL link to access your Node-RED Starter application.

STEP 5: CONFIGURE YOUR NODE-RED APPLICATION

The first time you open your Node-RED app, you'll need to configure it and set up security.

- A new browser tab will open with the Node-RED start page.
- Configure Node-RED app
- On the initial screen, click Next to continue.
- Secure your Node-RED editor by providing a username and password. If you need to change these at any point, you can either edit the values in the Cloudant database, or override them using environment variables. The documentation on nodered.org describes how to do this. Click Next to continue.
- The final screen summarizes the options you've made and highlights the environment variables you can use to change the options in the future. Click Finish to proceed.
- Node-RED will save your changes and then load the main application. From here you can click the Go to your Node-RED flow editor button to open the editor.

STEP 6: ADD EXTRA NODES TO YOUR NODE-RED PALETTE

Node-RED provides the palette manager feature that allows you to install additional nodes directly from the browser-based editor. This is convenient for trying nodes out, but it can cause issues due to the limited memory of the default Node-RED starter application.

The recommended approach is to edit your application's package.json file to include the additional node modules and then redeploy the application.

This step shows how to do that in order to add the node-red-dashboard module.

- On your application's details page, click the url in the Continuous Delivery box. This will take you to a git repository where you can edit the application source code from your browser.
- Scroll down the list of files and click on package.json. This file lists the module dependencies of your application.
- Click the **Edit** button
- Add the following entry to the top of the dependencies section:
- At this point, the Continuous Delivery pipeline will automatically run to build and deploy that change into your application. If you view the Delivery Pipeline you can watch its progress. The Build section shows you the last commit made and the Deploy section shows the progress of redeploying the application.
- Once the Deploy stage completes, your application will have restarted and now have the node-red-dashboard nodes preinstalled.

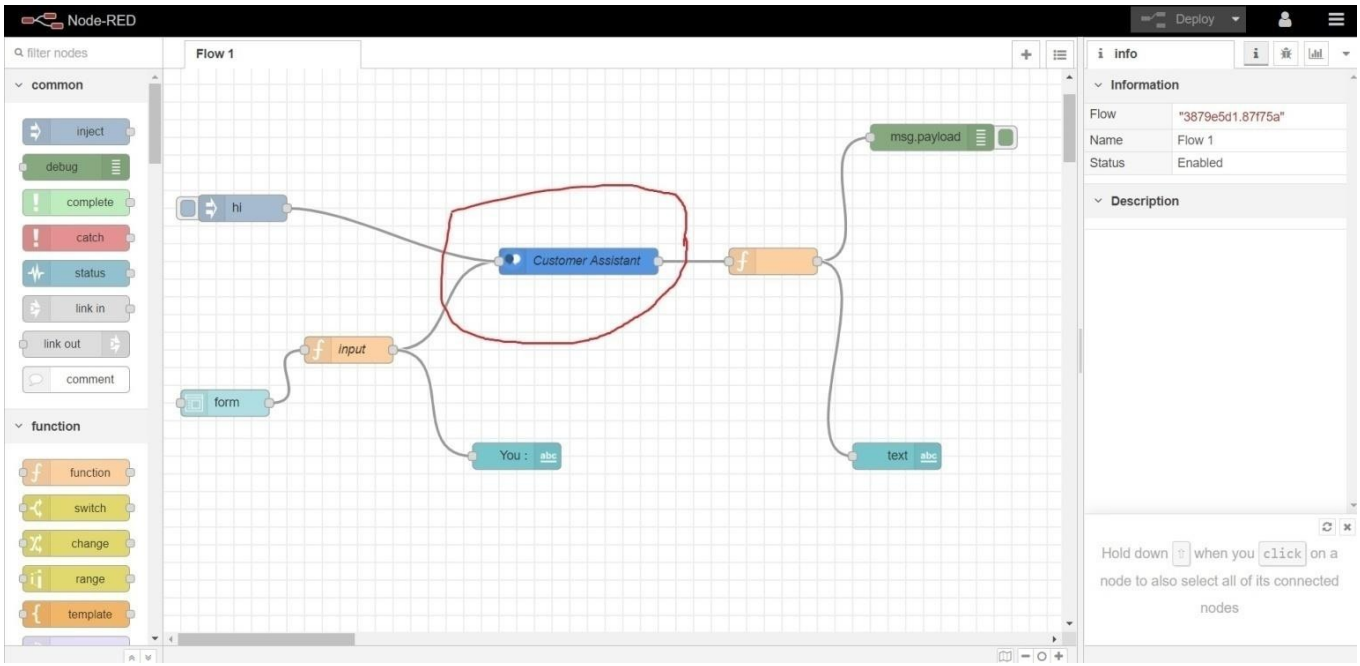
iv) INTEGRATION OF WATSON ASSISTANT IN NODE-RED

- 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 inject node on to the flow from the Input section
- Drag Debug on to the flow from the output section
- Double-click on the inject node
- Select the payload as a string
- Enter a sample input to be sent to the assistant service and click on done

- Enter a sample input to be sent to the assistant service and click on done

Connect the nodes as shown below and click on Deploy

- Open Debug window as shown below
- Click on the button to send input text to the assistant node
  - Observe the output from the assistant service node
  - The Result output is located inside “output.text"
- Drag the function node to parse the JSON data and get the bot response
- Double click on the function node and enter the JSON parsing code as shown below and click on done
  - Connect the nodes as shown below and click on Deploy

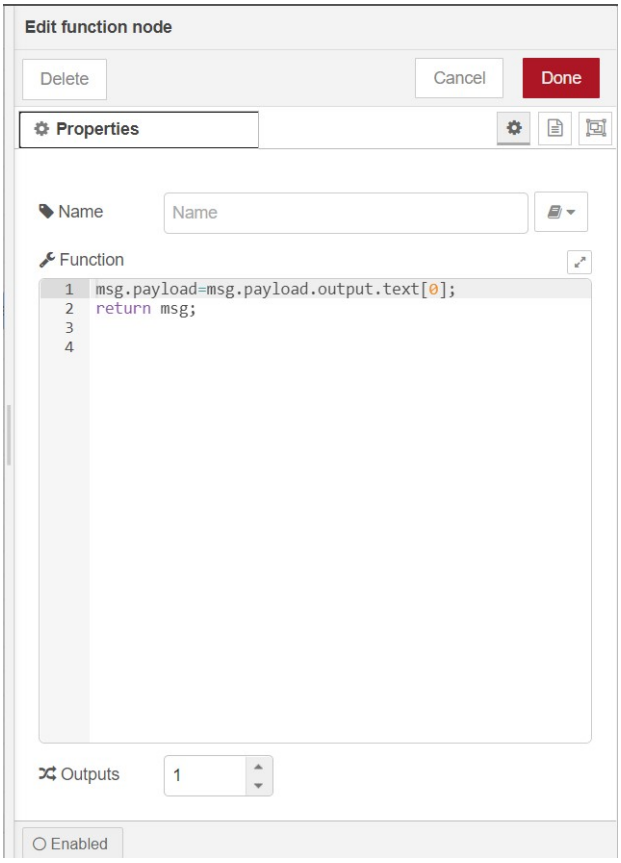


- Re-inject the flow and observe the parsed output

We are done integrating Watson assistant service to Node-red. In the next lab, we will create a web application using Node-red for the chatbot. 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

- Doube click on the “form” node to configure
- Click on the edit button to add the “Group” name and “Tab” name
- Click on the edit button to add tab name to web application
- Give sample tab name and click on add do the same thing for the group
- Give the label as “Enter your input”, Name as “text” and click on Done
- Drag a function node, double-click on it and enter the input parsing code as shown below

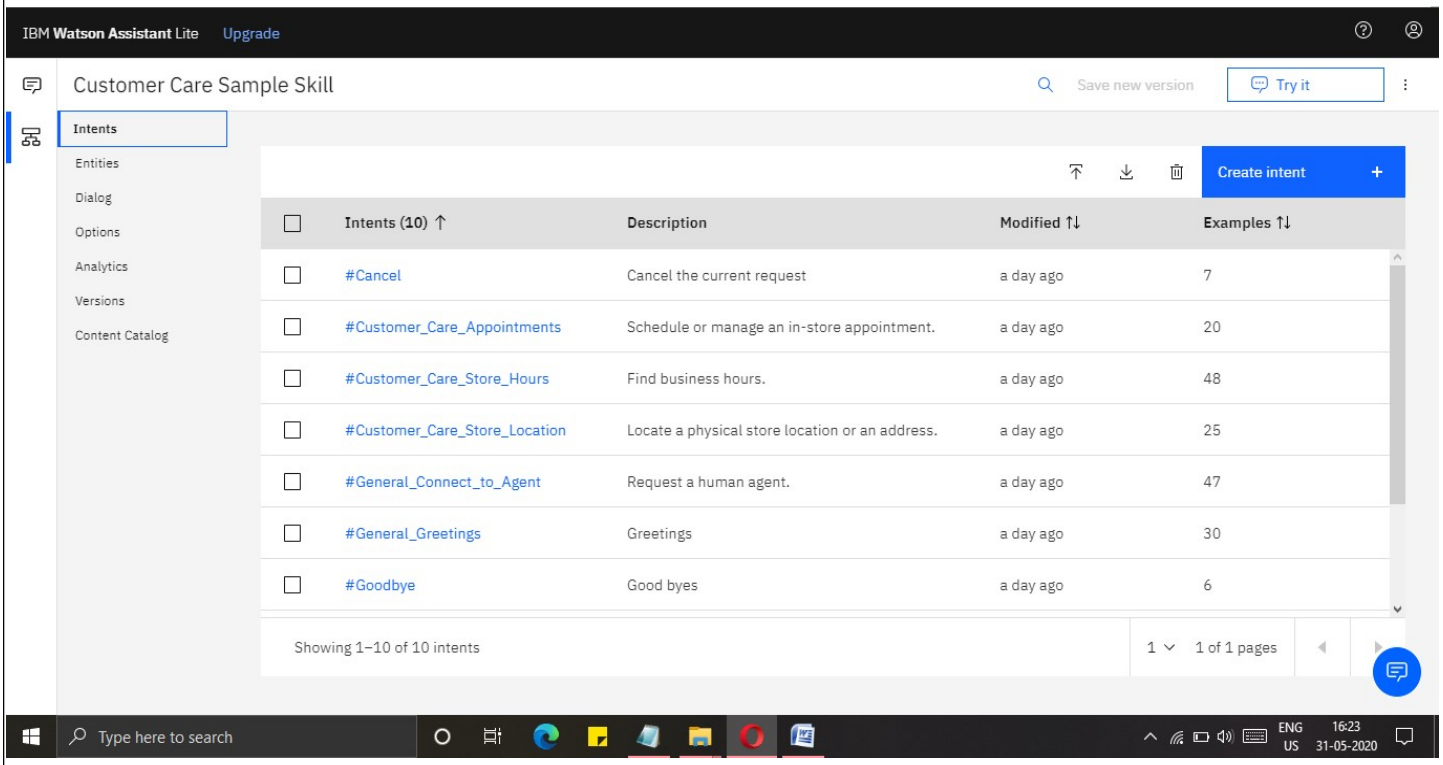


- Click on done
- Connect the form output to the input of the function node and output of the function to input of assistant node
- Search for “text” node from the “dashboard” section
- Drag two “text” nodes on to the flow
- Double click on the first text node, change the label as “You” and click on Done
- Double click on the second text node, change the label as “Bot” and click on Done
- Connect the output of “input parsing” function node to “ You” text node and output of “Parsing” function node to the input of “Bot” text node
- Click on Deploy

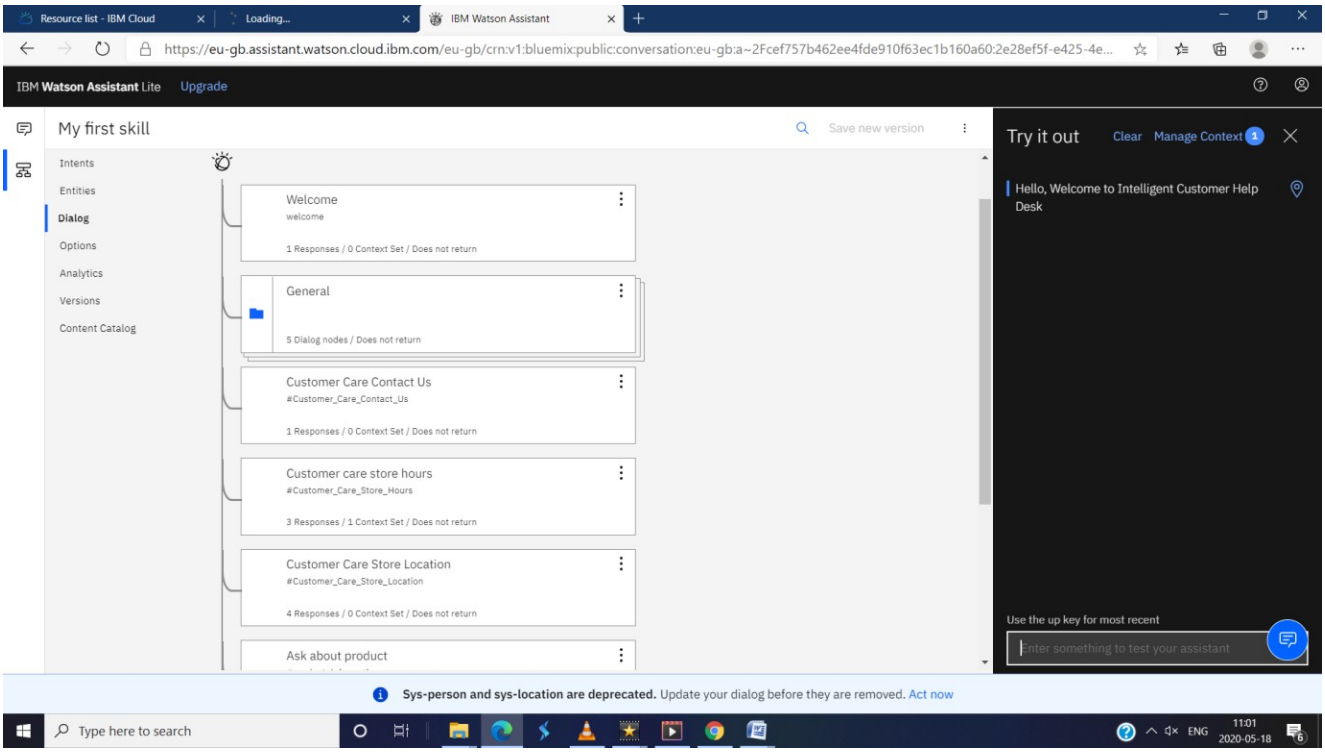
SCREENSHOTS

1. IBM WATSON ASSISANT:

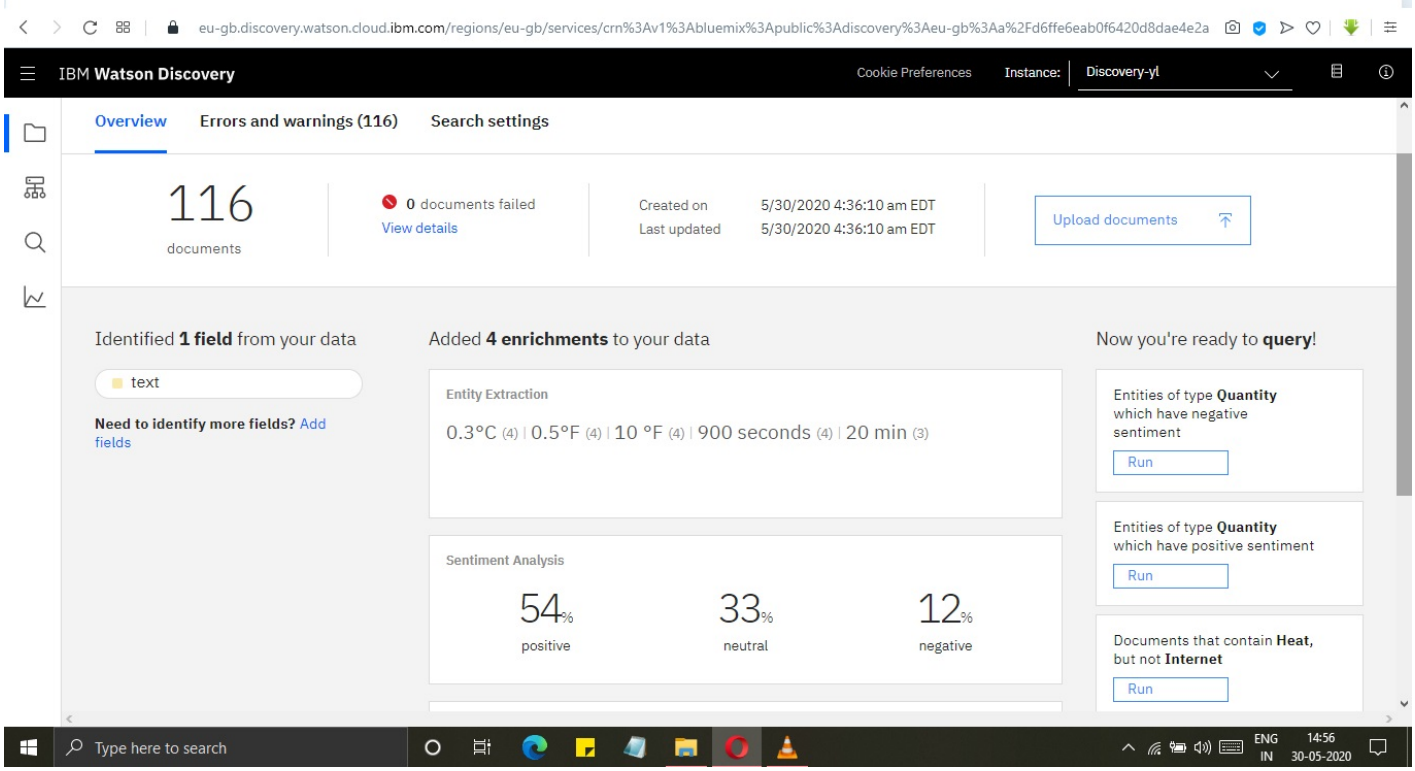
Intent Page :

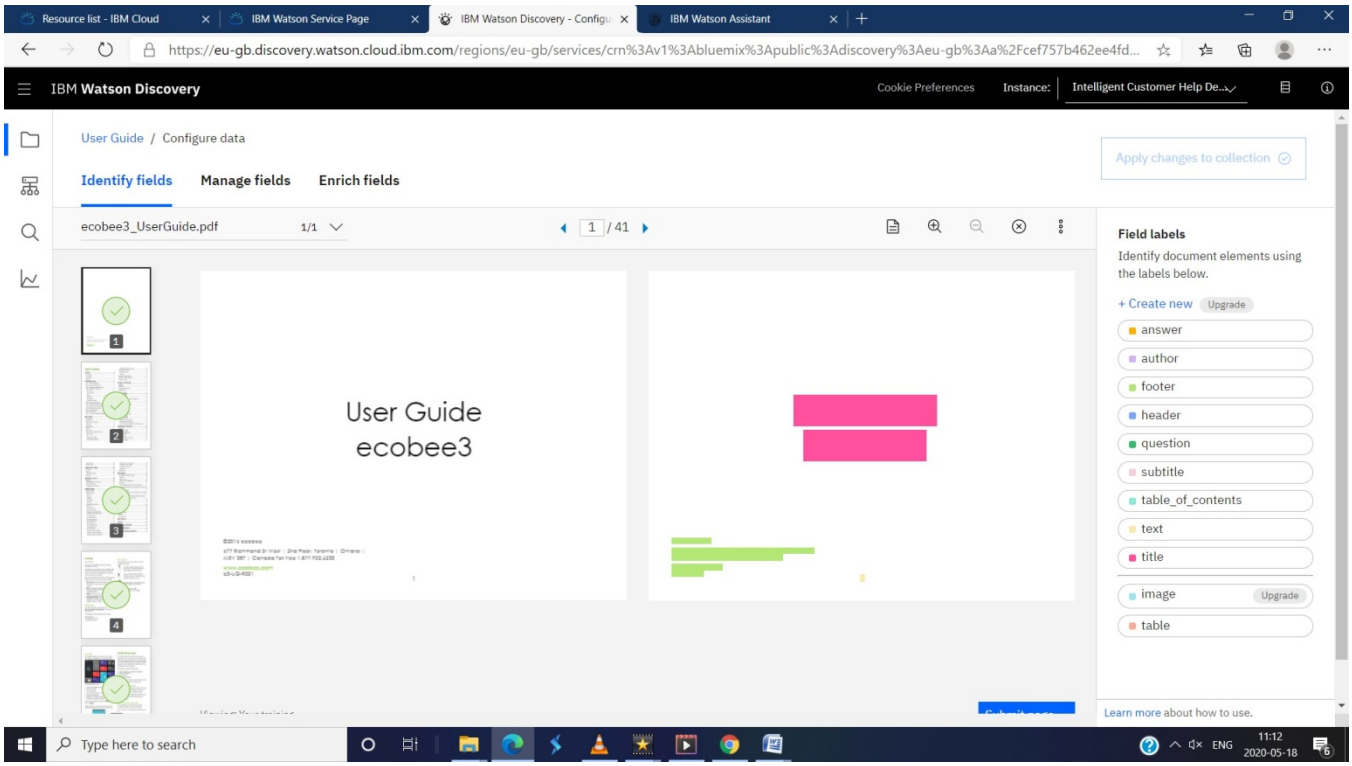


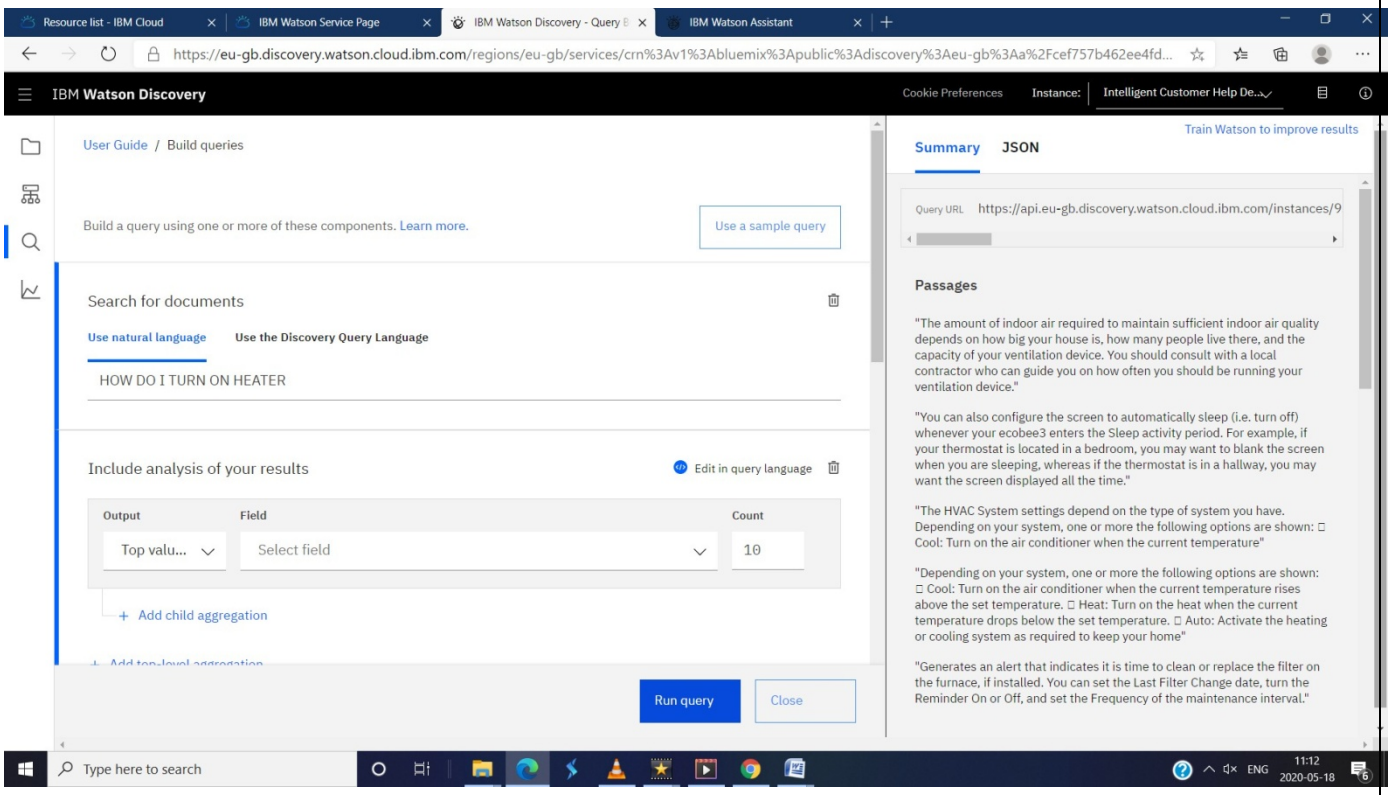
Dialog Page:



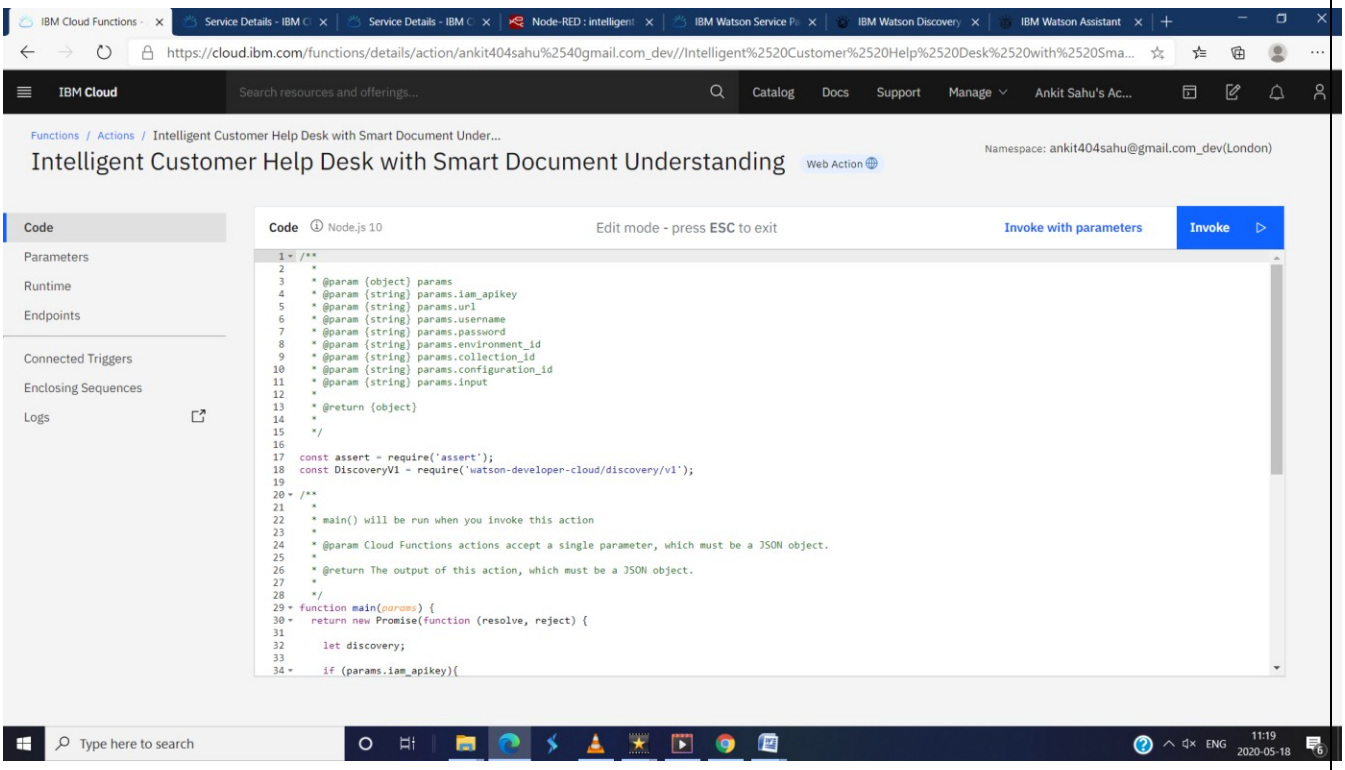
2. IBM WATSON DISCOVERY:



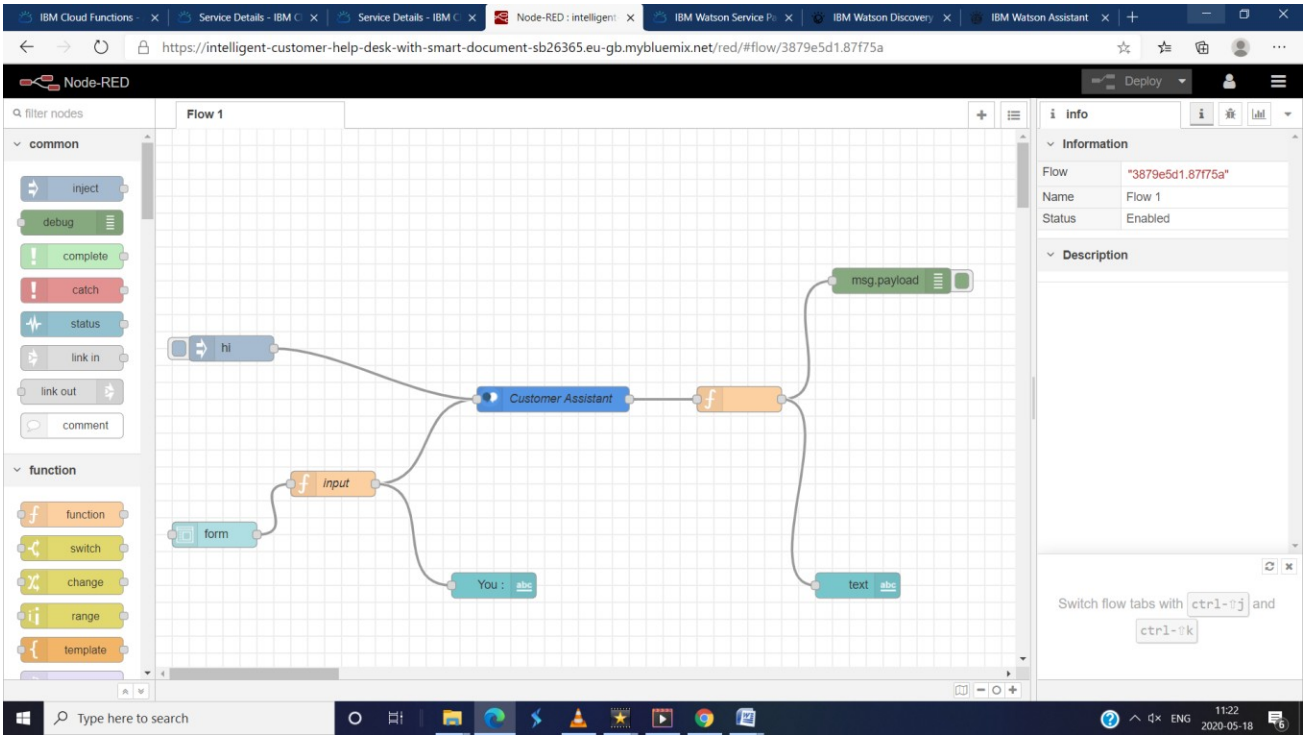




3. CLOUD FUNCTION:

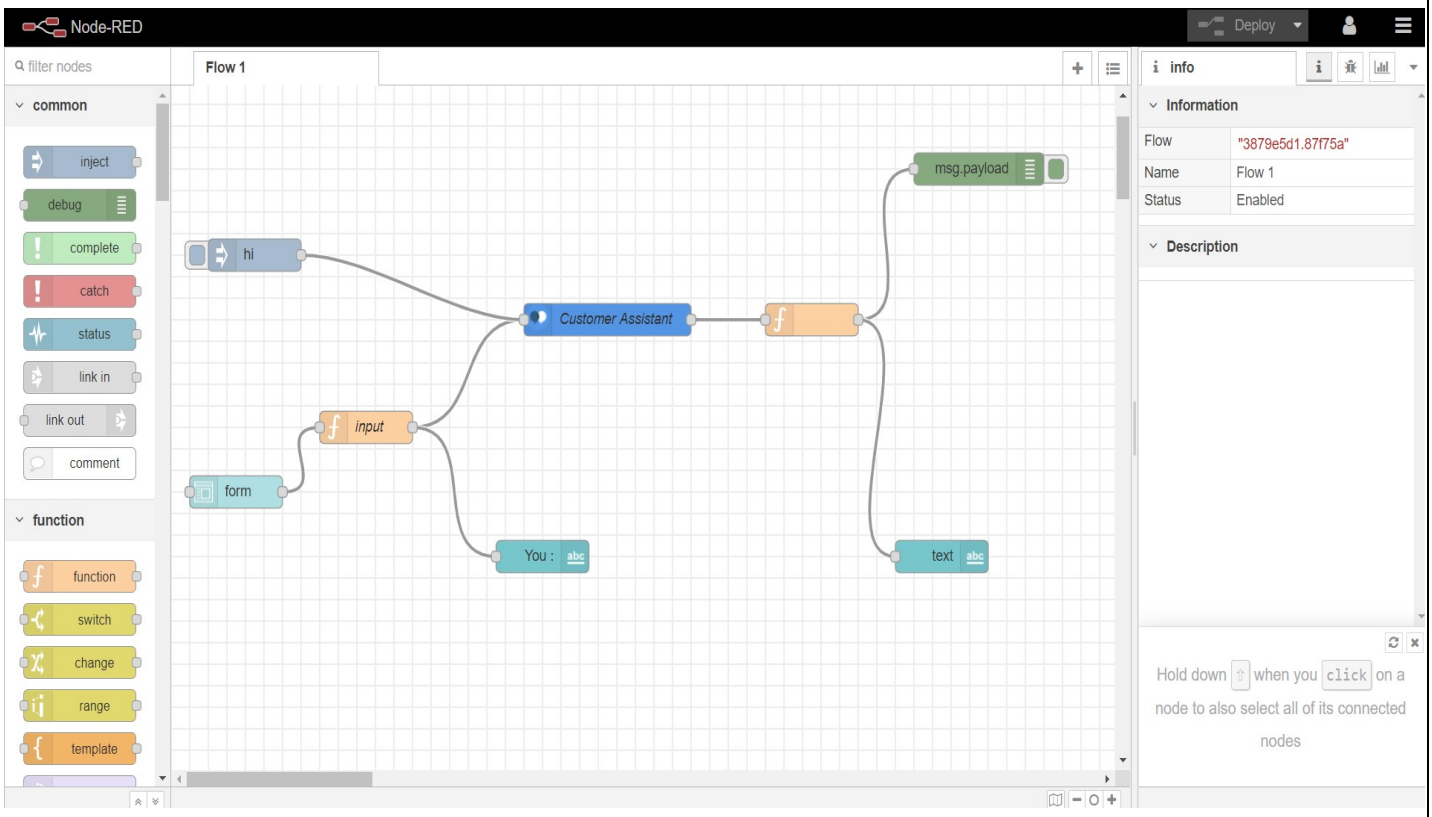


4. NODE RED:


IBM Cloud Functions
Service Details - IBM C
Service Details - IBM C
Node-RED : intelligent
IBM Watson Service Pa
IBM Watson Discovery
IBM Watson Assistant
https://intelligent-customer-help-desk-with-smart-document-sb26365.eu-gb.mybluemix.net/red/#flow/3879e5d1.87f75a
Node-RED
Deploy
filter nodes
Flow 1
common
inject
debug
complete
catch
status
link in
link out
comment
function
function
switch
change
range
template
hi
form
input
Customer Assistant
msg.payload
You :
text
info
Information
Flow
"3879e5d1.87f75a"
Name
Flow 1
Status
Enabled
Description
Switch flow tabs with ctrl-⇧j and ctrl-⇧k
Type here to search
ENG
11:22
2020-05-18

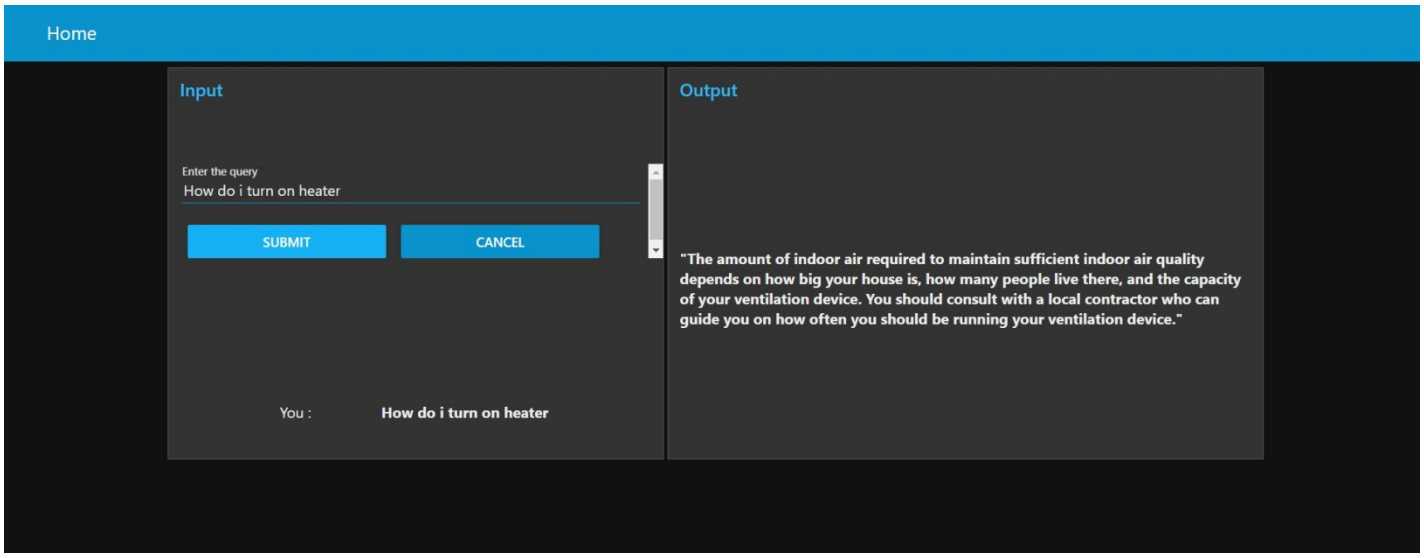
5. **FLOWCHART**

- Create flow and configure all nodes:
- 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



6. RESULTS

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



ADVANTAGES & DISADVANTAGES

Advandtages :

- 365X24X7 Support — Unlike your support staff, Chatbots don't need any breaks, they happily work and learn 24 hours a day. They have a robust cloud architecture.
- Reduce operational and service expense
- Get a new age platform to wow your customers
- Increase engagement with customers and touchpoints
- Eliminate mobile app-fatigue
- Multiply reach, increase breadth and depth of engagement
- Rich analytics and customer interaction
- Instantaneous response without the need for human response delays

Disadvantages :

- Lack Emotions
- It is very challenging to create a chatbot from scratch. It requires that you invest significant time and effort into creating it. You may also need to have some coding knowledge to create a better-functioning chatbot.
- They have been designed to handle first-level questions only. They may not be able to solve complex queries.
- Chatbots require ongoing review, maintenance, and optimization in terms of their knowledge base and the way they are supposed to communicate with our customers.

7. APPLICATIONS

- It can deploy in popular social media applications like facebook massenger, slack, telegram etc.
- This chatbot can deploy any website to clarify basic doubts of viewers.

8. CONCLUSION

By doing the above procedure and all we successfully created Intelligent helpdesk smart chartbot using IBM Watson assistant, Watson discovery, cloud-functions and Node-RED service.

9. FUTURE SCOPE

We can include watson studio text to speech and speech to text services to access the chatbot handsfree. This is one of the future scope of this project.

10. BIBILOGRAPHY

- https://cloud.ibm.com/docs

APPENDIX

Source code

1. **Watson Assistant**
   - **My_first_skill.json :**
     This code is too long. So I put this code on the Github.

2. **Watson Discovery**

   Here I have used " **ecobee3_UserGuide.pdf"** to train the Watson discovery.

3. **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);
   });
  });
}
```

4. **Node Red**

- **Flow.json :**

[{"id":"f3891dc0.4d097","type":"function","z":"3879e5d1.87f75a","name":"","func":"msg.payload=msg.payload.output.text[0];\nreturn msg;\n\n","outputs":1,"noerr":0,"x":690,"y":320,"wires":[["1a31d881.e44577","881e2391.d752d"]]}