

## PROJECT REPORT

|                      |   |  |
|----------------------|---|--|
| <b>Project ID</b>    | : | SPS_PRO_99   |
| <b>Project Title</b> | : | Intelligent Customer Help Desk with Smart Document Understanding |
| <b>Category</b>      | : | Artificial Intelligence  |

|                     |   |              |
|---------------------|---|--------------|
| <b>Name</b>         | : | Kalgee Kotak |
| <b>SMARTINTERNZ</b> |   |              |



## **Table of Contents**

- 1. INTRODUCTION**
  - 1.1. Overview**
  - 1.2. Purpose**
- 2. LITERATURE SURVEY**
  - 2.1. Existing Problem**
  - 2.2. Proposed Solution**
- 3. THEORETICAL ANALYSIS**
  - 3.1. Block Diagram**
  - 3.2. Hardware/Software Designing**
- 4. EXPERIMENTAL INVESTIGATIONS**
- 5. FLOWCHART**
- 6. RESULT**
- 7. ADVANTAGES AND DISADVANTAGES**
- 8. APPLICATIONS**
- 9. CONCLUSION**
- 10. FUTURE SCOPE**
- 11. BIBLIOGRAPHY**
- APPENDIX**
  - A. Source Code**

# 1. INTRODUCTION

## 1.1 Overview

### Project Summary

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.

**Project Requirements:** Python, IBM Cloud, IBM Watson

**Functional Requirements:** IBM CCloud

**Technical Requirements:** AI, ML, PYTHON, WATSONAI

**Software Requirements:** Watson assistant

**Project Deliverables:** SmartInterz Internship

**Project Team:** Kalgee Kotak

**Project Schedule:** No. of Days: 30 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

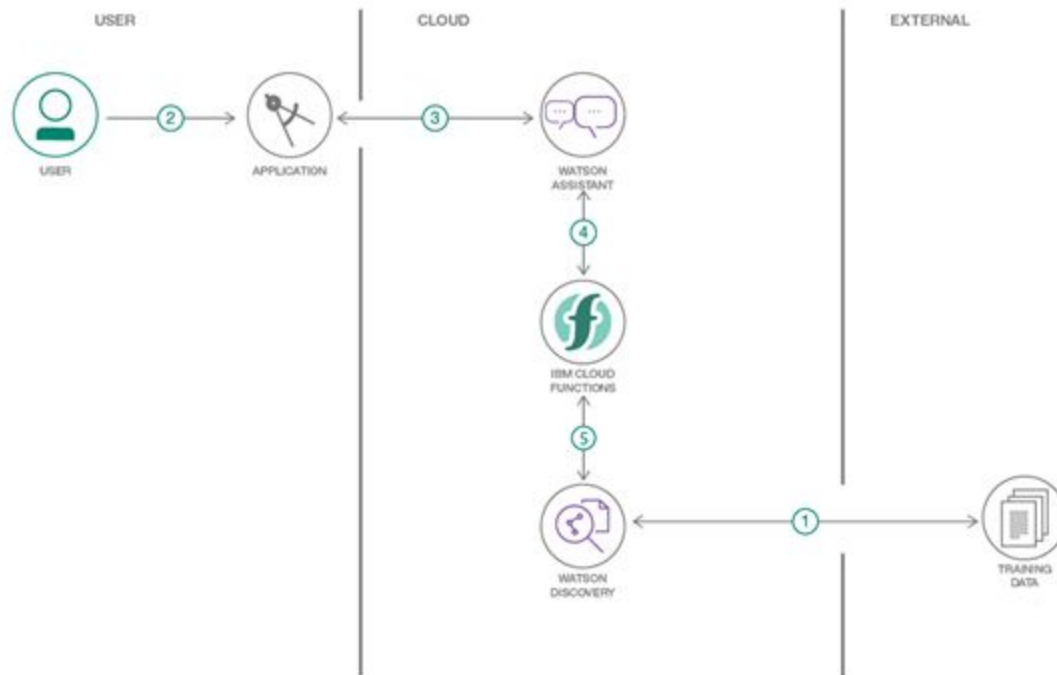
The typical customer care chatbot can answer simple questions, such as store locations and hours, directions, and maybe even making appointments. When a question falls outside of the scope of the predetermined question set, the option is typically to tell the customer the question isn't valid or offer to speak to a real person.

### 2.2. Proposed Solution

In this project, there will be another option. If the customer question outside of scope of the predetermined question set, the application shall pass the question onto Watson Discovery Service, which has been pre-loaded with the ecobee3 manual. So now, instead of "Would you like to speak to a customer representative?" we can return relevant sections of the manual to help solve customer's problems.

### 3. THEORETICAL ANALYSIS

#### 3.1. Block 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.

## 3.2. Hardware/Software Designing

The software utilized are Watson Discovery, Watson Assistant, and Node-Red.

**Watson Discovery**: Discovery makes it possible to rapidly build cognitive, cloud-based exploration applications that unlock actionable insights hidden in unstructured data. It applies the latest breakthroughs in machine learning, including natural language processing capabilities, and is easily trained on the language of your domain. It breaks open the data silos and retrieves specific answers to user's questions while analysing trends and relationships buried in the uploaded data. With Smart Document Understanding (SDU) feature you can train IBM Watson Discovery to extract custom fields in your documents.

**Watson Assistant**: It is a conversational AI platform that provides customers fast, straightforward and accurate answers to their questions, across any application, device, or channel. By addressing common customer inquiries, Watson Assistant reduces the cost of customer interactions. It uses Watson's AI machine learning (ML) and natural language understanding (NLU). The user input is received by the assistant and routes it to a dialog skill. The dialog skill interprets the input further, then directs the flow of the conversation. The dialog gathers any information that needs to respond to the user's behalf.

**Node-Red**: It is a prototyping tool that builds applications easily. Applications are developed by building data flows through a series of connected nodes. Intricate applications can be built that allow Watson services to interact with a range of capabilities and services exposed as Node-Red nodes. It helps in wiring together hardware devices, APIs, and online services without writing any code. It provides a web-based flow editor, which can be used to connect these services and also create UI.

## 4. EXPERIMENTAL INVESTIGATIONS

Steps :

### a. Create IBM Cloud Services

Create the following services:

- [Watson Discovery](#)
- [Watson Assistant](#)

### b. Configure WatsonDiscovery

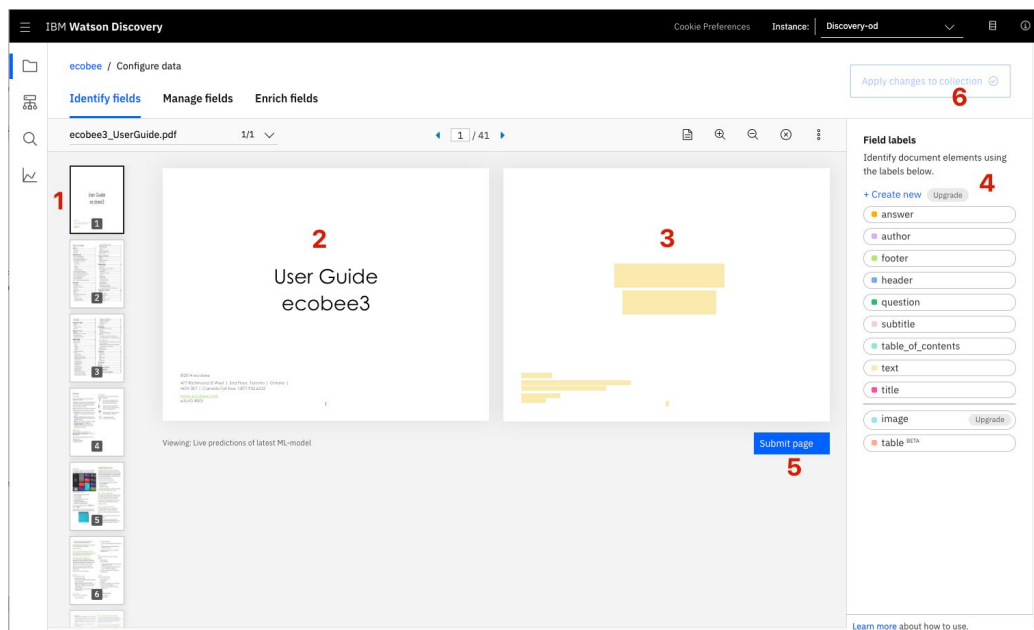
#### Import the document

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.

The Ecobee is a popular residential thermostat that has a wifi interface and multiple configuration options.

#### Annotate with SDU

Now let's apply SDU to our document to see if we can generate some better query responses. From the Discovery collection panel, click the Configure data button (located in the top right corner) to start the SDU process. Here is the layout of the Identify fields tab of the SDU annotation panel:



The goal is to annotate all of the pages in the document so Discovery can learn what text is important, and what text can be ignored.

- [1] is the list of pages in the manual. As each is processed, a green check mark will appear on the page.
- [2] is the current page being annotated.
- [3] is where you select text and assign it a label.
- [4] is the list of labels you can assign to the page text.
- Click [5] to submit the page to Discovery.
- Click [6] when you have completed the annotation process.

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

For this specific owner's manual, at a minimum, it is suggested to mark the following:

- The main title page as title
- The table of contents (shown in the first few pages) as table\_of\_contents
- All headers and sub-headers (typed in light green text) as a subtitle
- All page numbers as footers
- All warranty and licensing information (located in the last few pages) as a footer
- All other text should be marked as text.

Once you click the Apply changes to collection button, you will be asked to reload the document. Choose the same owner's manual .pdf document as before.

Next, click on the Manage fields tab.

- Tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except subtitles and text.
- Select a "Subtitle" as field to split the document apart
- Submit your changes.

Once again, you will be asked to reload the document.

Now, as a result of splitting the document apart, your collection will look very different: Return to the query panel (click Build your own query).

### **Store credentials for future use**

In upcoming steps, you will need to provide the credentials to access your Discovery collection, so store them.



### c. Create IBM Cloud Functionsaction.

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 [2], cut and paste in the code provided. The code is pretty straight-forward - it simply connects to the Discovery service, makes a query against the collection, then returns the response.

Select the Parameters tab and 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. 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.

## d. Configure WatsonAssistant.

Launch the Watson Assistant tool and create a new dialog skill. Select the Use sample skill option as your starting point. This dialog skill contains all of the nodes needed to have a typical call center conversation with a user.

### **Add new intent**

The default customer care dialog does not have a way to deal with any questions involving outside resources, so we will need to add this. Create a new intent that can detect when the user is asking about operating the Ecobee thermostat. From the Customer Care Sample Skill panel, select the Intents tab. Click the Create intent button. Name the intent #Product\_Information, and at a minimum, enter the following example questions to be associated with it.

### **Create new dialog node**

Now we need to add a node to handle our intent. Click on the Dialog tab, then click on the drop down menu for the Small Talk node , and select the Add node below option. Name the node "Ask about product" and assign it our new intent . This means that if Watson Assistant recognizes a user input such as "how do I set the time?", it will direct the conversation to this node.

### **Enable webhook from Assistant**

Set up access to our WebHook for the IBM Cloud Functions action you created earlier. Select the Options tab. Enter the public URL endpoint for your action. Return to the Dialog tab, and click on the Ask about product node. From the details panel for the node, click on Customize, and enable Webhooks for this node. Click Apply. The dialog node should have a Return variable set automatically to \$webhook\_result\_1. This is the variable name you can use to access the result from the Discovery service query. You will also need to pass in the users question via the parameter input . The key needs to be set to the value:

"<?input.text?>"

If you fail to do this, Discovery will return results based on a blank query.

### 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 turn on heater?" has triggered our dialog node, which is indicated by the #Product\_Informatio 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.

### Get IBM Cloud services credentials and add to .env file

Update the .env file with the credentials from your Assistant service.

```
# Copy this file to .env and replace the credentials with  
# your own before starting the app.
```

```
# Watson Assistant  
ASSIATANT_URL=<add_assistant_url>  
ASSISTANT_APIKEY=<add_assistant_apikey>  
ASSISTANT_SKILL_ID=<add_assistant_skill_id>
```

```
# Run locally on a non-default port (default is 3000)  
# PORT=3000
```

Credentials can be found by clicking the Service Credentials tab, then the View Credentials option from the panel of your created Watson service.

## e. Create a Node red flow to connect all the services together.

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


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.

### **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 and then selecting Resource List . You will see your newly created Node-RED Application listed under the Apps section . You will also see a corresponding entry under the Cloud Foundry apps section . 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 Cloudfant database, or override them using environment variables. The documentation on [nodered.org](https://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 .INTEGRATION OF WATSON ASSISTANT IN NODERED**

Drag assistant v1 node on to the flow. Double-click on the Watson assistant node. Give a name to your node and enter the username, password and assistant 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. Connect the nodes 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 and click on done. Connect the nodes below and click on Deploy. Re-inject the flow and observe the parsed output. We are done integrating Watson assistant service to Node-red.

## f. Create flow and configure node in order to develop web application for chatbot

In order to create a web application UI we need "dashboard" nodes which should be installed manually. Go to navigation pane and click on manage palette. Click on install. Search for "node-red-dashboard" and click on install and again click on install on the prompt. The following message indicates dashboard nodes are installed, close the manage palette. Search for "Form" node and drag on to the flow. Double click on the "form" node to configure. Click on the edit button to add the "Group" name and "Tab" name. Click on the edit button to add tab name to web application. Give sample tab name and click on add. Do the same thing for 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

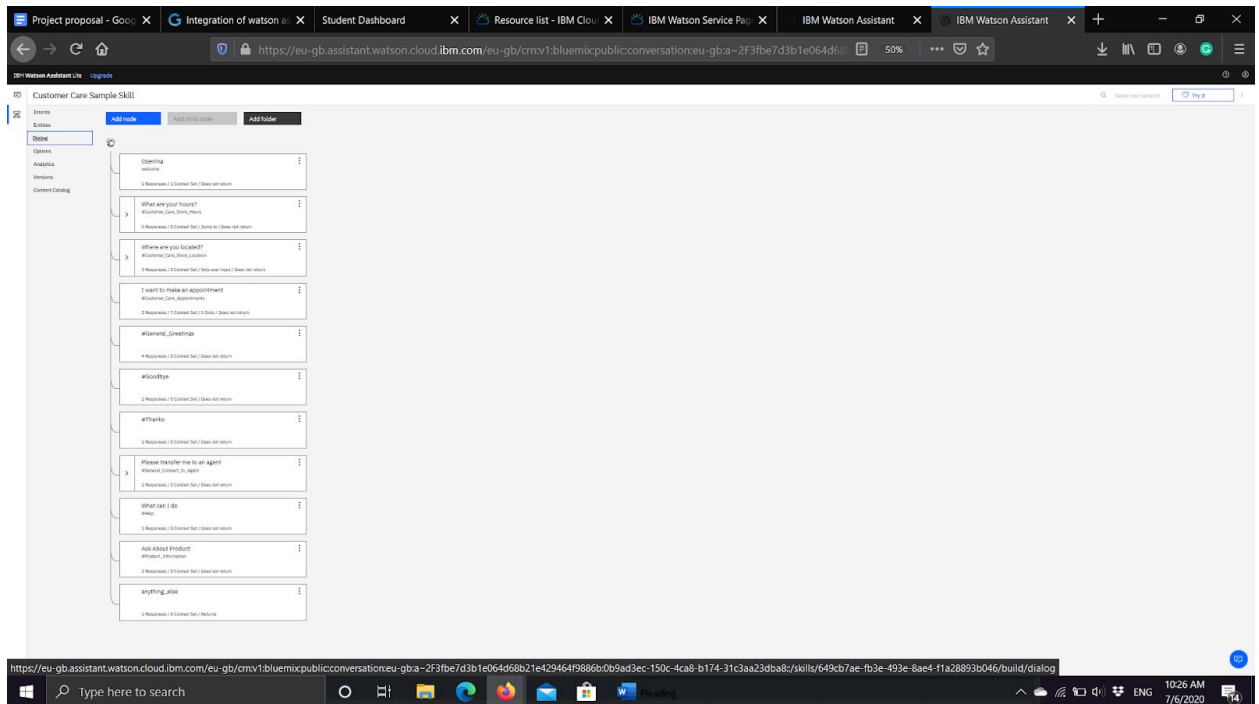
- Watson Discovery – to understand the document:

The screenshot shows the IBM Watson Discovery interface. At the top, there's a navigation bar with tabs for 'Student Dashboard', 'Integration of wat...', 'Build Chatbot using...', 'IBM Cloud Function...', 'IBM Watson Service...', 'IBM Watson Assis...', 'IBM Watson Discov...', and 'IBM/watson-discov...'. The main header displays the URL 'https://eu-gb.discovery.watson.cloud.ibm.com/regions/eu-gb/services/cm%3A%2F%3Apublic%3Adiscovery%3A...' and the instance name 'Discovery-g'. Below the header, the 'Overview' tab is selected, showing '117 documents'. A summary section indicates '0 documents failed' and provides creation and last updated timestamps. The 'Identified 5 fields from your data' section lists 'footer', 'subtitle', 'table\_of\_contents', 'text', and 'title'. The 'Added 4 enrichments to your data' section shows 'Entity Extraction' with results like '0.3°C', '0.5°F', '10 °F', '900 seconds', and '20 min'. It also displays 'Sentiment Analysis' with 56% positive, 31% neutral, and 14% negative sentiment. 'Concept Tagging' results include 'Heat', 'HVAC', 'Internet', 'Heat pump', and 'Thermodynamics'. A 'Category Classification' section shows 'technology and com... operating systems'. On the right, there are buttons to 'Run' queries for 'Most common entity types', 'Documents that contain Heat, but not Internet', and 'Top people related to /technology and computing/operating systems'.

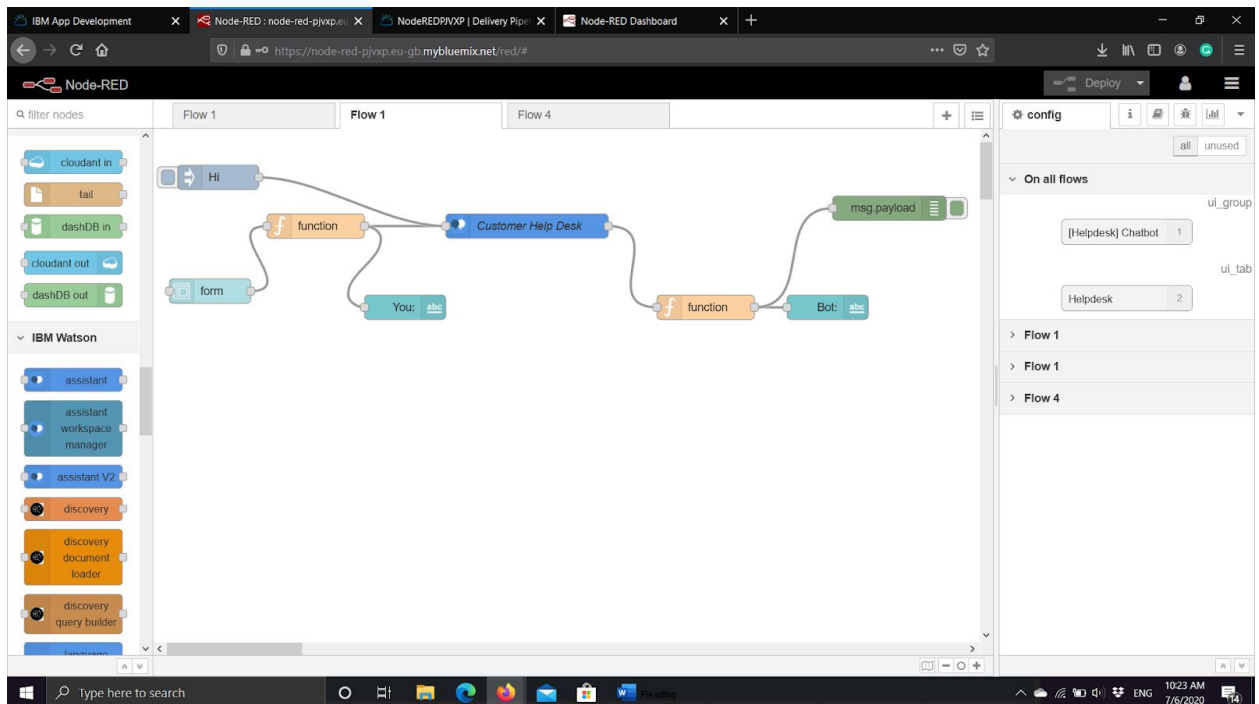
- Cloud Function – action generation:

The screenshot shows the IBM Cloud Functions console. The top navigation bar includes tabs for 'Student Dashboard', 'Integration of wat...', 'Build Chatbot using...', 'IBM Cloud Function...', 'IBM Watson Service...', 'IBM Watson Assis...', 'IBM Watson Discov...', and 'IBM/watson-discov...'. The main header displays the URL 'https://cloud.ibm.com/functions/details/action/kalgee.kotak%2540gmail.com\_dev/disco-action-1/code' and the namespace 'kalgee.kotak@gmail.com\_dev(London)'. The 'disco-action-1' function is selected, showing its 'Code' tab. The code is a Node.js function that uses the Watson Discovery API to analyze documents. The 'Activations' section shows a single activation with an ID '5a68355b75864b75a8355b75868b754e' and a duration of 1058 ms. The 'Results' section displays the output of the function, including 'matching\_results', 'passages', and 'enriched\_text'.

- Watson Assistant – to build chatbot:

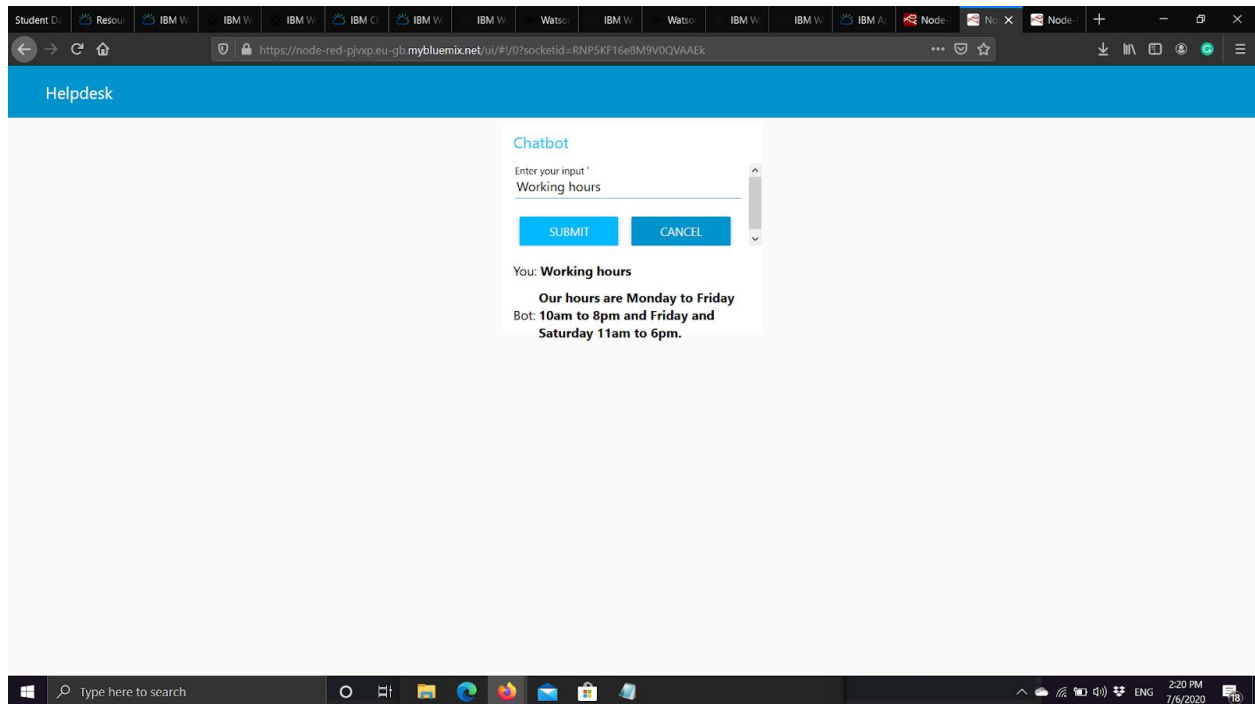


- Create cloud foundry app  
Link : <https://node-red-pjvxp.eu-gb.mybluemix.net/red/#>
- Node-Red flow – UI creation (Flow based tool):



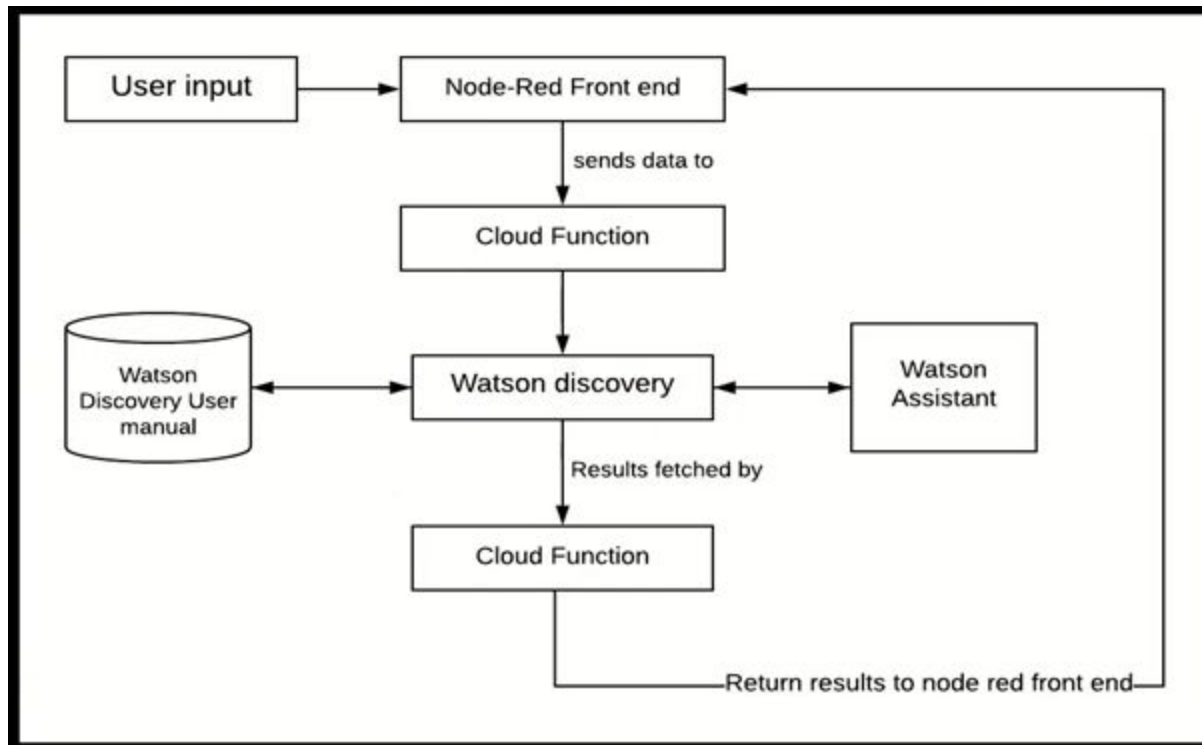


- After deploying the chatbot from dashboard UI can be seen.

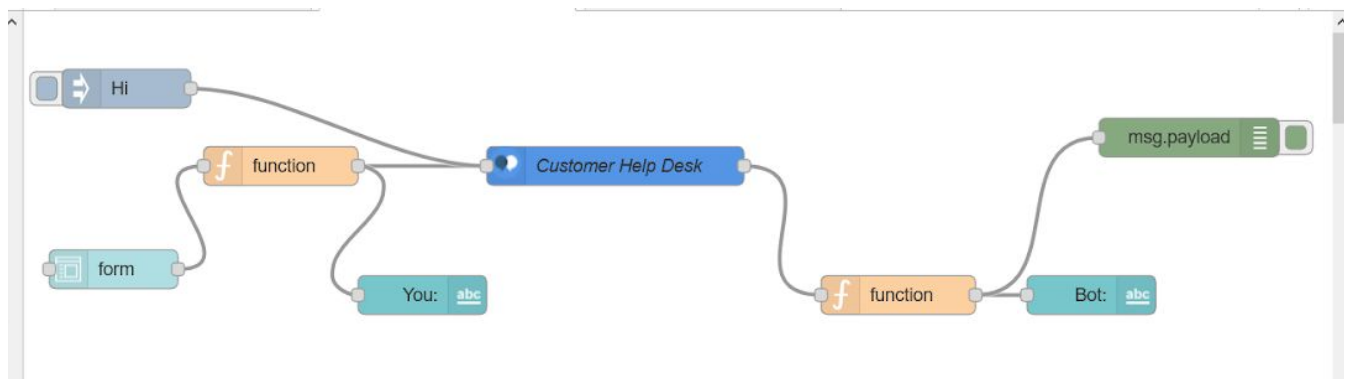



## 5. FLOWCHART

Flowchart of general flow of project:



## Node-Red Flow





The flow summarizes the functionality of the Chatbot. The document preloaded in Discovery is being annotated by the Smart Document Understanding feature. This trains the Discovery to improve the query result. The user interacts with frontend UI of the chatbot and it keeps them engaged in conversation with small talk. This interaction between the user and the UI is being coordinated by Watson Assistant.

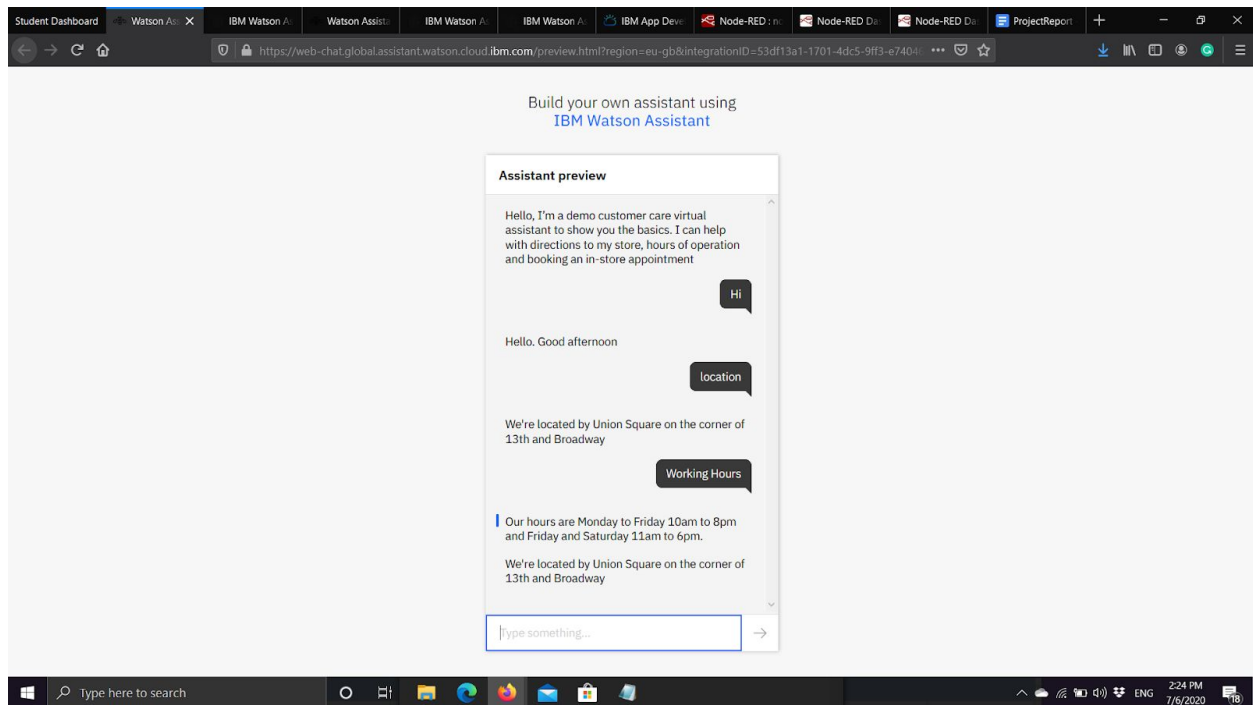
When a user asks a technical query, the Assistant invokes the Cloud Function action. This action then queries Watson Discovery and returns with the relevant results from the pre-loaded owner's manual.

The Watson Assistant, Discovery, and Cloud Functions are IBM services. The Training Data is the owner's manual which is an external section.

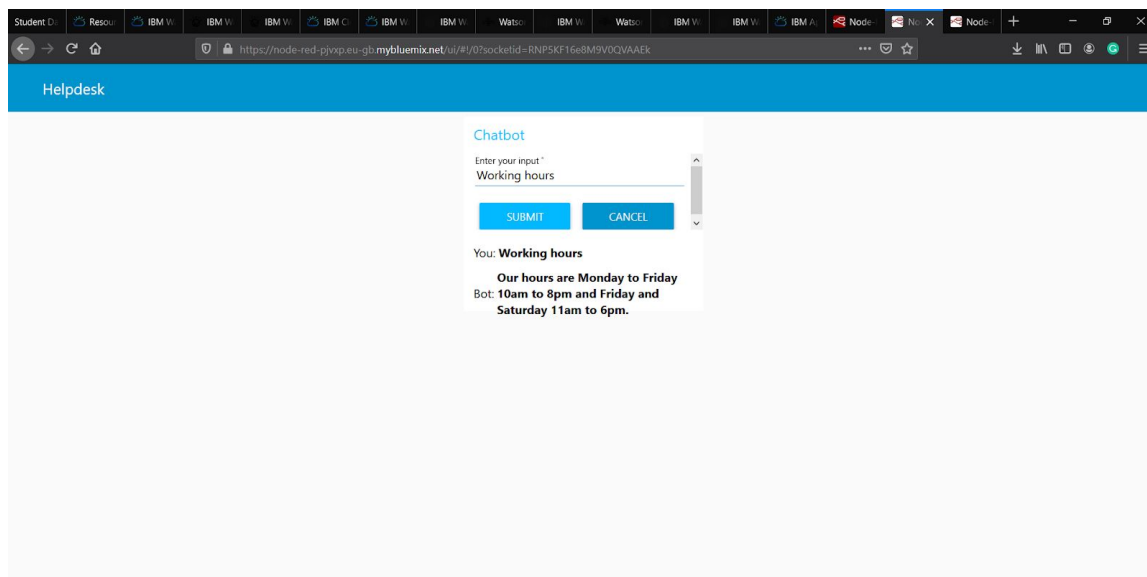
## 6. RESULT

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

### IBM Watson Assistant



### Node-Red Dashboard



## 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 the 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.

**4] Text-to-Speech and Speech-to-Text:** In future this project could include text to speech and speech to text nodes in the node red flow. This would enable the user to operate the bot hands-free. The node red flow of the bot could also include language translator nodes to cater the needs of a wide spectrum of users.

## 11. BIBLIOGRAPHY

1. [https://www.ibm.com/cloud/architecture/tutorials/cognitive\\_discovery](https://www.ibm.com/cloud/architecture/tutorials/cognitive_discovery)
2. <https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started>
3. <https://developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chatbot-on-nodered/>
4. <http://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red>
5. <https://github.com/IBM/watson-discovery-sdu-with-assistant>

## APPENDIX

### A. Source Code

#### 1. Cloud Functions

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

```

## 2. Watson Assistant

Please refer to the link:

<https://github.com/SmartPracticeschool/IIIPS-INT-2689-Intelligent-Customer-Help-Desk-with-Smart-Document-Understanding/blob/master/skill-Customer-Care-Sample-Skill.json>

## 3. Node Red

```

[{"id":"76af0565.b89e24","type":"tab","label":"Flow
1","disabled":false,"info":"","},{id":"f587f2d8.a9a45","type":"function","z":"76af0565.b89e24",
"name":"","func":"msg.payload= msg.payload.text;\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":200,"y":120,"wires":[["2c2e33ca.9c7
98c","ce7f0760.f4ed88"]]},{"id":"2d3ec4a3.16a164","type":"function","z":"76af0565.b89e24",
"name":"","func":"msg.payload=msg.payload.output.text[0];\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":680,"y":220,"wires":[["b54f0e8d.45f
8b","b3c0c58a.99d538"]]},{"id":"2c2e33ca.9c798c","type":"watson-conversation-v1","z":"76af
0565.b89e24","name":"Customer Help
Desk","workspaceid":"649cb7ae-fb3e-493e-8ae4-f1a28893b046","multiuser":false,"context":
true,"empty-payload":false,"service-endpoint":"https://api.eu-gb.assistant.watson.cloud.ibm
.com/instances/0b9ad3ec-150c-4ca8-b174-31c3aa23dba8","timeout":"","optout-learning":
false,"x":478,"y":103,"wires":[["2d3ec4a3.16a164"]]},{"id":"b3c0c58a.99d538","type":"debug"
,"z":"76af0565.b89e24","name":"","active":true,"tosidebar":true,"console":false,"tostatus":fal
se,"complete":"payload","targetType":"msg","statusVal":"","statusType":"auto","x":906,"y":98
,"wires":[[]]},{"id":"be483cd4.956358","type":"inject","z":"76af0565.b89e24","name":"","props":
[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"
topic":"","payload":"Hi","payloadType":"str","x":80,"y":60,"wires":[["2c2e33ca.9c798c"]]},{"id":
"19e988b6.0376f7","type":"ui_form","z":"76af0565.b89e24","name":"","label":"","group":"96

```

```
97ce2d.59d1a8","order":1,"width":"0","height":"0","options":[{"label":"Enter your
input","value":"text","type":"text","required":true,"rows":null}], "formValue":{"text":""},"paylo
ad":"","submit":"submit","cancel":"cancel","topic":"","x":70,"y":200,"wires":[["f587f2d8.a9a45
"]]],{"id":"ce7f0760.f4ed88","type":"ui_text","z":"76af0565.b89e24","group":"9697ce2d.59d1a
8","order":2,"width":"0","height":"0","name":"","label":"You:","format":{"msg.payload}}","lay
out":"row-left","x":310,"y":220,"wires":[]}, {"id":"b54f0e8d.45f8b","type":"ui_text","z":"76af056
5.b89e24","group":"9697ce2d.59d1a8","order":3,"width":"0","height":"0","name":"","label":"
Bot:","format":{"msg.payload}}","layout":"row-spread","x":830,"y":220,"wires":[]}, {"id":"9697
ce2d.59d1a8","type":"ui_group","z":"","name":"Chatbot","tab":"65f6572f.1889d","order":1,"di
sp":true,"width":"6","collapse":false}, {"id":"65f6572f.1889d","type":"ui_tab","z":"","name":"He
lpdesk","icon":"dashboard","disabled":false,"hidden":false}]
```