**Project Report**

**Internship title**: Intelligent Customer Help Desk with Smart Document Understanding - SB28734

**Project ID:** SPS\_PRO\_99

**Project Title :**

Intelligent Customer Help Desk with Smart Document Understanding

**Introduction :**

The project takes an informative tour through the IBM services. Smart document understanding is the part of Natural Language Processing that comes under the machine learning domain of dealing with the documents and text data. Here with the help of IBM services we propose a system that is capable of understanding the user query through a chatbot platform and looks for an appropriate answer from the relevant document.

**Watson Discovery** is the IBM cloud service that helps infuse AI into the organization's business. With IBM Watson Discovery, you can ingest, normalize, enrich, and search your unstructured data (JSON, HTML, PDF, Word, and more) with speed and accuracy. It packages core Watson APIs such as Natural Language Understanding and Document Conversion along with UI tools that enable you to easily upload, enrich, and index large collections of private or public data.

**Watson Assistant** is the platform that we use as the conversational medium through which the user can query and the chatbot responds appropriately. Watson Assistant is a conversation AI platform that helps you provide 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, helping your agents focus on complex use cases – not repetitive responses.

**IBM Cloud Functions** is a distributed computing service that executes application logic in response to requests from web or mobile apps. You can set up specific actions to occur based on HTTP-based API requests from web apps or mobile apps, and from event-based requests from services like Cloudant. We use this cloud service to connect the discovery service to our Watson assistant. The request to our assistant is forwarded to the discovery service through this cloud functions. Watson discovery responds in the JSON format. The answer with the highest possibility of the occurrence is extracted and shown as a chatbot response to the user.

**Node-RED** is built on Node.js, taking full advantage of its event-driven, non-blocking model. We use the node-red application in order to connect our services to the web interface. Requests and responses are carried out through the HTTP mechanism. Using node-red we create a REST API for our services which is a stateless service on the server-side. Using node-red we provide a path to our services to interact with each other.

**Webhooks** is a mechanism that allows you to call out to an external program based on events in your dialog. Using this we can specify the request URL for an external API we want to be able to invoke from the dialog node. Watson Assistant will call this URL when configured to do so.

**Proposed System :**

> We use the above services to meet our project goal i.e. when the user queries to the chatbot, cloud function takes that query to the discovery service where the most likely response is generated and the response is sent back to the user through the Watson assistant platform.

> Here for the illustrative purpose, we take a user manual of sensor equipment from Ecobee. We give this document to the discovery service. Initially, the service requires us to help service gain insight into our data i.e. to help service identify whether the piece of text is a title or subtitle or header etcetera. After this, the service trains on its own giving us ready to use the trained model.

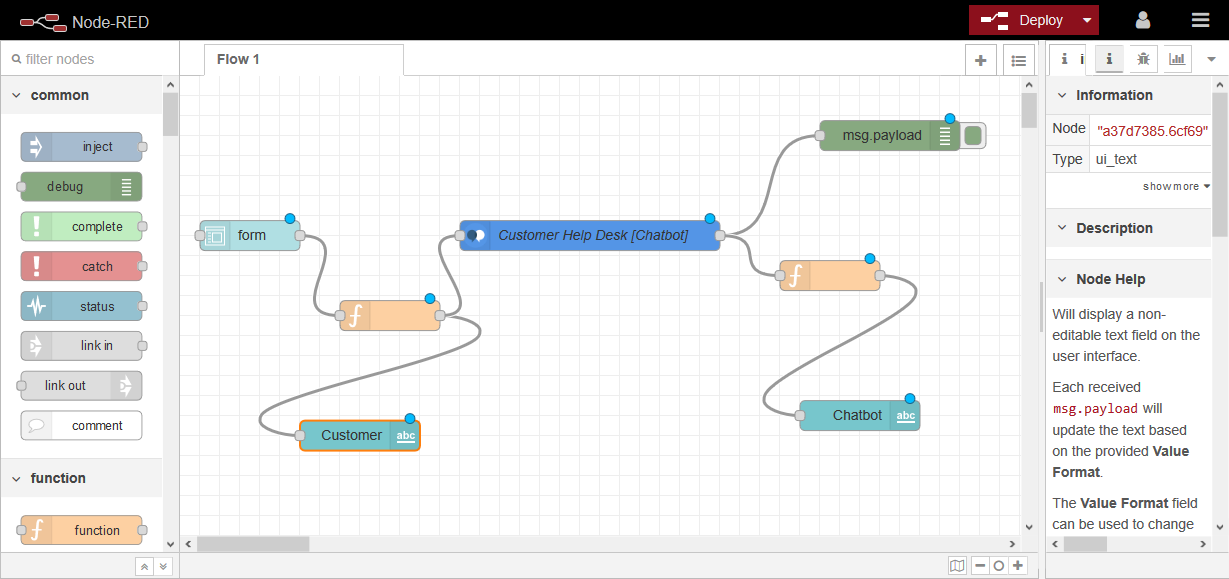
> We develop Watson assistant as our chatbot platform to interact with the user. The chatbot is given permission to the web interface through the node-red application.

In the final stages, we test our chatbot with the illustrative query. The web interface is developed with a simple form structure to display the user query and respond to that query.

\*\* Further details are discussed in the demo video.

**Implementation and results.**

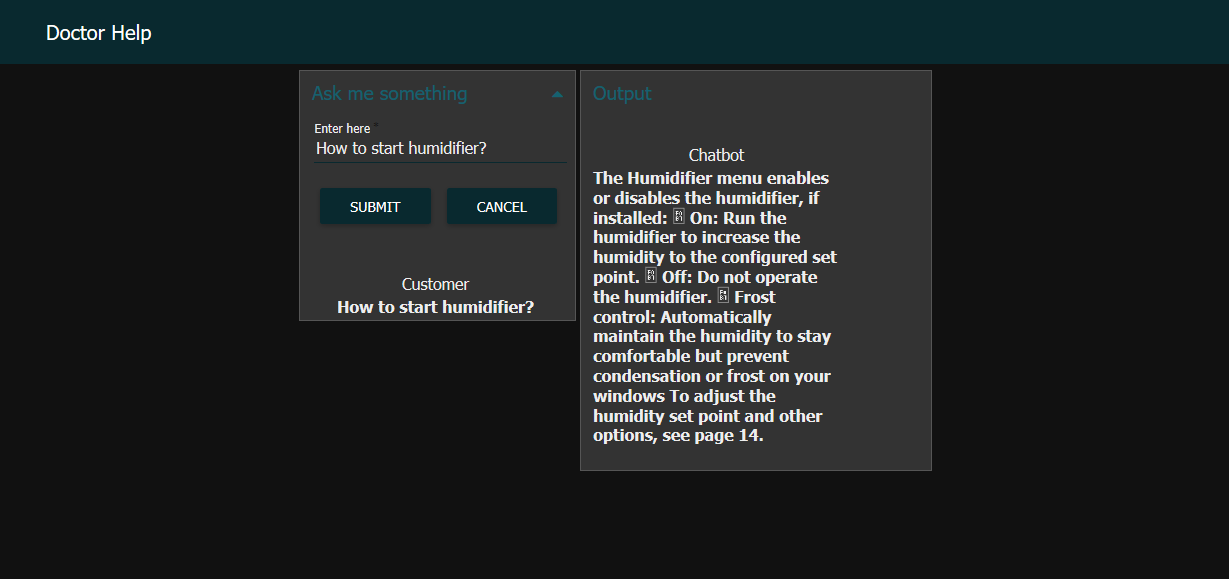
Below is the node-RED flow.



> As can be seen from the snippet, our assistant is connected to the form element through function element which takes user input from the input element.

> After processing of user query assistant returns the response in the form msg.payload which is shown as the output on the web interface.

> Msg.payload carries the input user query to the assistant service and also brings the output from assistant to the web interface.



**Project links :**

Chatbot (web interface) link - https://node-red-xmxdm.eu-gb.mybluemix.net/ui/#!/0?socketid=j44ahmSXBTAnxizOAAAf

Github link - https://github.com/SmartPracticeschool/llSPS-INT-764-Intelligent-Customer-Help-Desk-with-Smart-Document-Understanding

Video link-

**References :**

https://www.ibm.com/cloud/get-started

https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/

<https://www.youtube.com/watch?v=hitUOFNne14>

<https://www.youtube.com/embed/5z3i5IsBVnk>

<https://www.youtube.com/embed/Jpr3wVH3FVA>

https://cloud.ibm.com/docs/services/discovery?topic=discovery-getting-started

https://developer.ibm.com/articles/introduction-watson-discovery/

https://developer.ibm.com/technologies/web-development/articles/ws-restful

https://cloud.ibm.com/docs/openwhisk?topic=cloud-functions-getting-starte

<https://www.youtube.com/embed/G3bqRndQtQg>

<https://www.youtube.com/embed/SLwpqD8n3d0>