**Intelligent Customer Help Desk with**

**Smart Document Understanding**

A project report

by

W. SHIVANI PATNAIK

**Category**: Artificial Intelligence

**Application ID**: SPS\_APL\_20200003546

**Internship Title**: Intelligent Customer Help Desk with Smart Document Understanding - SB39130

**Project ID**: SPS\_PRO\_99

Internship at TheSmartBridge

**May 2020**

Email id: [shivanideepi27@gmail.com](mailto:shivanideepi27@gmail.com)

Github link: <https://github.com/SmartPracticeschool/llSPS-INT-1773-Intelligent-Customer-Help-Desk-with-Smart-Document-Understanding>

# **Table of Contents**

1. Introduction

1.1 Overview

1.2 Purpose

2. Literature Survey

2.1 Existing System Study

2.2 Proposed System

3. Theoretical Analysis

3.1 Block Diagram

3.2 Hardware/Software designing

4. Experimental Investigations

5. Flowchart

6. Result

7. Advantages & Disadvantages

8. Applications

9. Future Scope

10. Conclusion

11. Bibliography

Appendix

1. Source Code

# **1.INTRODUCTION**

# **Overview**

## Project Summary:

|  |  |
| --- | --- |
|  | * *The typical customer care chatbot can answer simple questions. When a question falls out 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 out of default question set****, the application shall* ***pass*** *the question onto* ***Watson Discovery Service****, which has been pre-loaded with the device’s owner’s manual. So, instead of default answer, we can return relevant sections of the owner’s manual to help solve our customers problems.* * *The project shall use the* ***Smart Document Understanding feature*** *of Watson Discovery to train it on what text in the owner’s manual is important.* * *This will improve the answers returned from the queries.* |

## Project Requirements:

|  |  |
| --- | --- |
|  | * **Functional** requirements: Intelligent customer help desk with SMART DOCUMENT understanding. * **Technical** requirements: Python, IBM Cloud, IBM Watson * **Hardware** requirements:  1. Processor: i5 8th gen or higher 2. Speed: 2GHz or more 3. Hard disk space: 8 Gb or more  * **Software** requirements:  1. Operating system: Windows 10 2. Browser: any 3. Spyder |

## Project Deliverables

|  |  |
| --- | --- |
|  | * *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.* * *Project Documentation* |

# **1.2 Purpose**

#### The typical customer care chatbot can answer simple questions, such as store locations and hours, directions, and maybe even making appointments. When a question falls outside of the scope of the pre-determined question set, 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. So unless and until customer speciﬁcally asks for a customer representative the bot will try to solve all your queries.

#### To take it a step further, the project shall use the Smart Document Understanding feature of Watson Discovery to train it on what text in the owners manual is important and what is not. This will improve the answers returned from the queries. Then using Watson actions as webhook, Watson Discovery can be integrated with Watson assistant. Finally using Node-Red, Watson assistant can be integrated with a web UI. This UI can then be used to connect with Watson assistant and chat with it.

# **2.Literature Survey**

# **2.1 Existing System Study**

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.

# **2.2Proposed System**

The proposed solution is a chatbot that is built using IBM Cloud Services. It is a web application that utilizes multiple IBM Watson services to create a better customer care experience. Using the Watson Discovery Smart Document Understanding (SDU) feature, we will enhance the Discovery model so that queries will be better focused to only search the most relevant information found in a typical owner's manual. Using Watson Assistant, we will use a standard customer care dialog to handle a typical conversation between a customer and a company representative. When a customer question involves operation of a product, the Assistant dialog will communicate with the Discovery service using a webhook. The webhook will be created by deﬁning a web action using IBM Cloud Functions.

## Smart Document Understanding (SDU)

SDU trains Watson Discovery to extract custom ﬁelds in your documents. Customizing how your documents are indexed into Discovery will improve

the answers returned from queries. With SDU, you annotate ﬁelds within your documents to train custom conversion models. As you annotate, Watson is learning and will start predicting annotations. SDU models can also be exported and used on other collections. Current document type support for SDU is based on your plan.

Lite plans: PDF, Word, PowerPoint, Excel, JSON, HTML.

Advanced plans: PDF, Word, PowerPoint, Excel, PNG, TIFF, JPG, JSON, HTML.

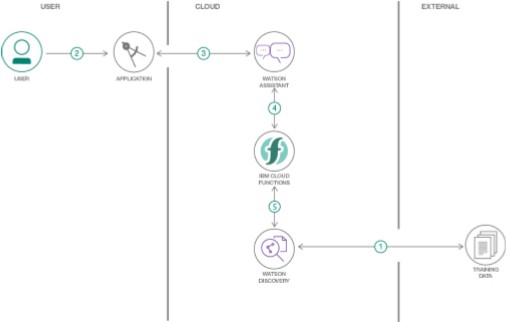
## Webhook

A webhook is a mechanism that allows you to call out to an external program based on something happening in your program. When used in a Watson Assistant dialog skill, a webhook is triggered when the Assistant processes a node that has a webhook enabled. The webhook collects data that you specify or that you collect from the user during the conversation and save in context variables, and sends the data to the Webhook request URL as an HTTP POST request. The URL that receives the webhook is the listener. It performs a predeﬁned action using the information that is provided by the webhook as speciﬁed in the webhook deﬁnition, and can optionally return a response.

In the proposed system, the webhook will communicate with an IBM Cloud Functions web action, which is connected to the Watson Discovery service.

# **3.Theoretical Analysis**

# **3.1 Block Diagram**



## **3.2 Hardware/Software designing**

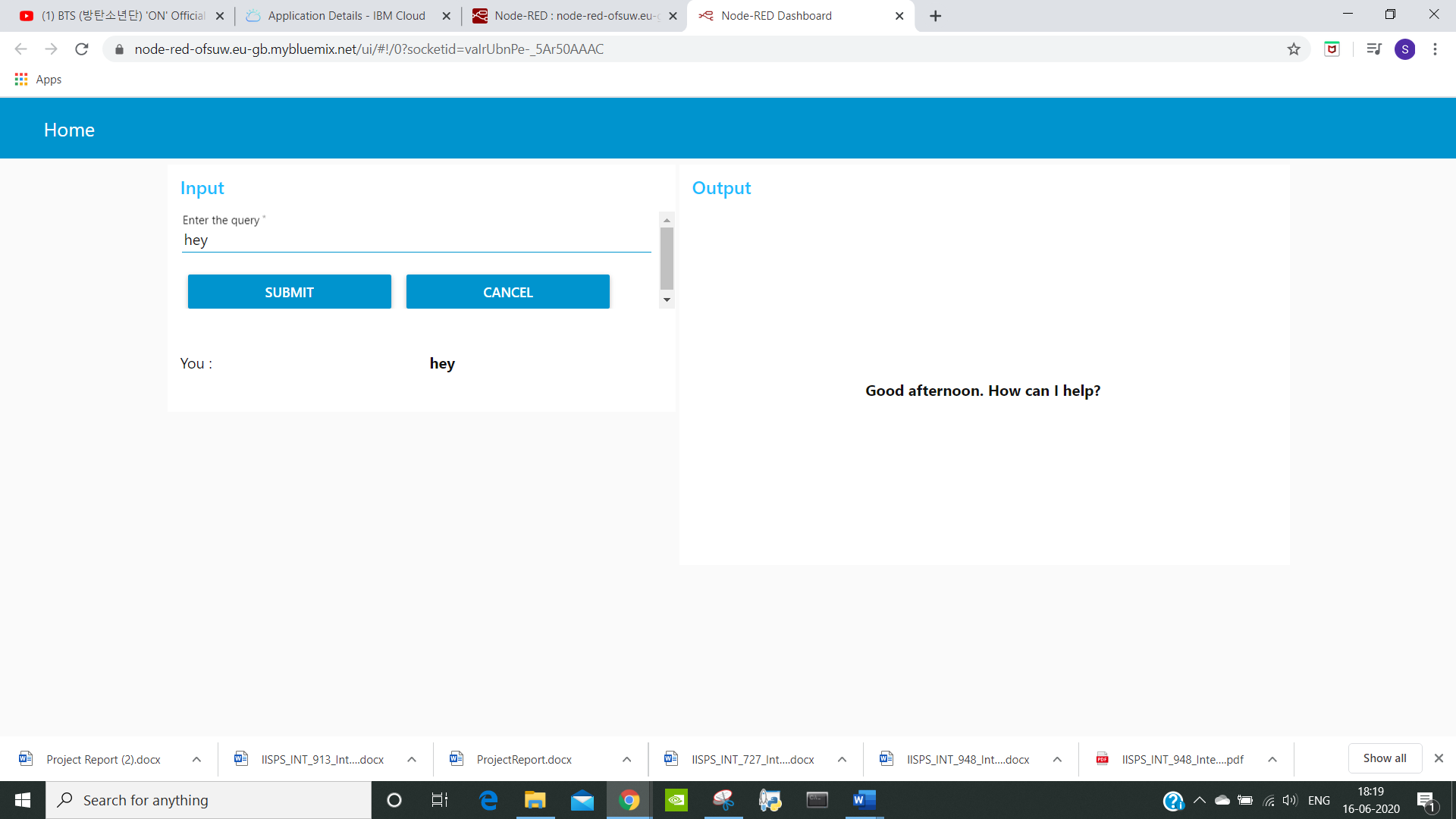
* Create necessary Watson services.
* Create a customer care dialog skill in Watson Assistant
* Use Smart Document Understanding to build an enhanced Watson Discovery collection
* Create an IBM Cloud Functions web action that allows Watson Assistant to post queries to Watson Discovery
* Build a web application with integration to all these services & deploy the same on IBM Cloud Platform.

# **4. Experimental Investigations**

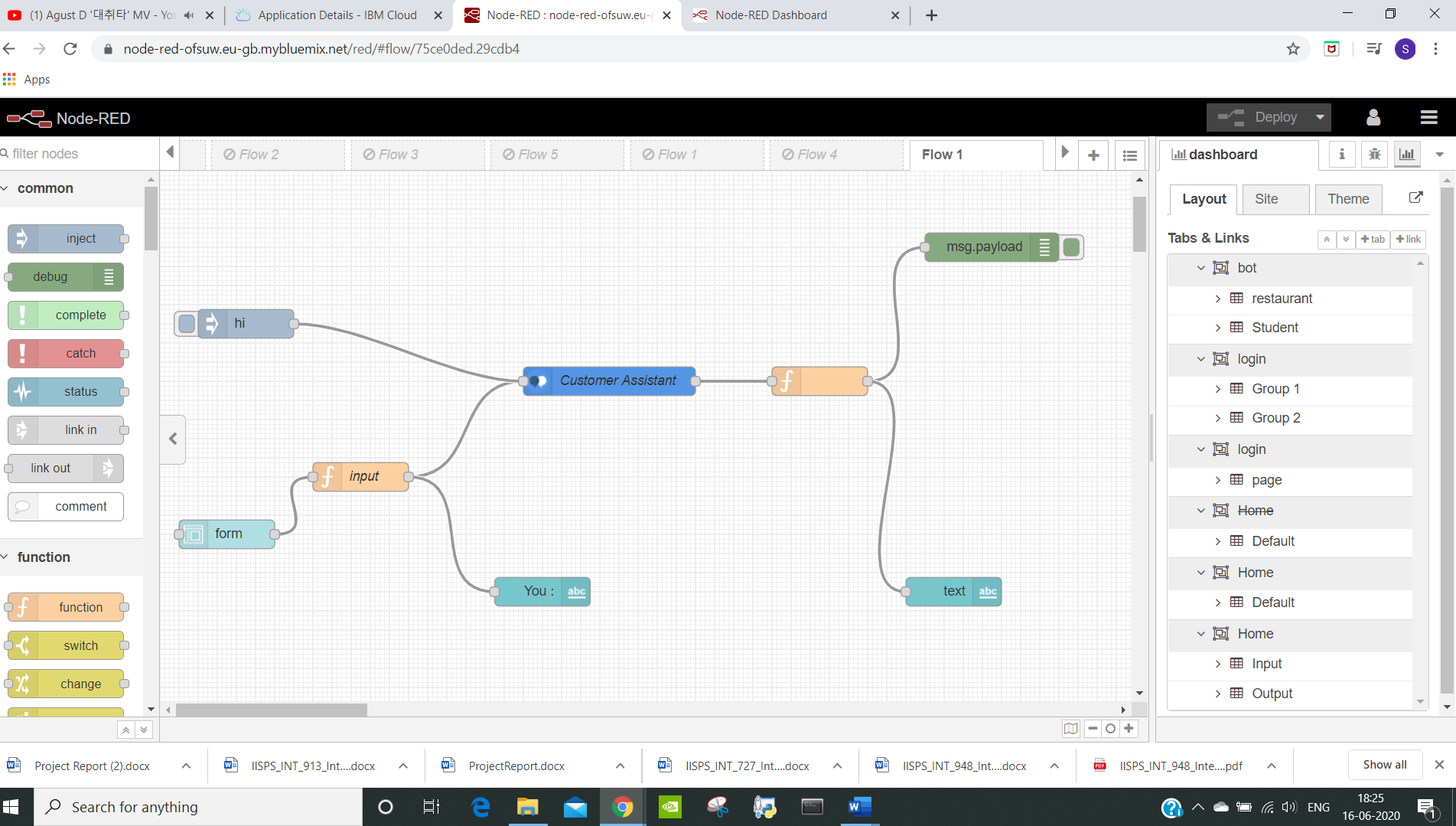
Create the following services:

* Watson Discovery
* Watson Assistant
* Node Red
* IBM cloud function

After the creation of all these services we ﬁnally integrate it in Node-Red App and obtain the following results:



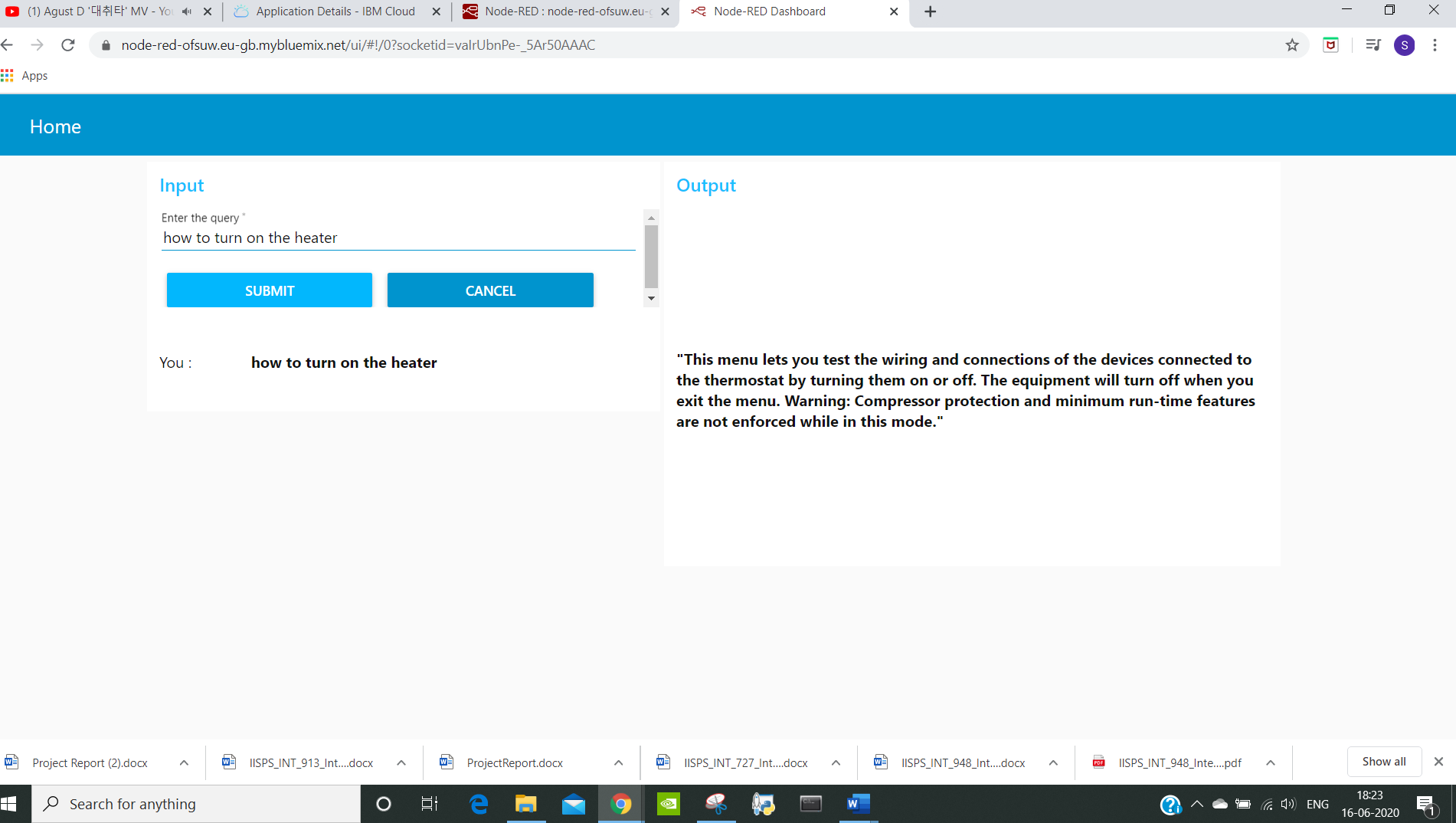
# **5. Flowchart**



# **6. Result**

#### Web based UI was developed by integrating all the services using Node-RED.

URL for the above : <https://node-red-ofsuw.eu-gb.mybluemix.net/ui/#!/0?socketid=vaIrUbnPe-_5Ar50AAAC>



# **7. Advantages & Disadvantages**

# Advantages:

* Increased customer satisfaction.
* Lower labour costs.
* Variety of uses.
* 24x7 availability.
* Multiple customer handling.
* Can be integrated with various external services.

# Disadvantages:

* Limited responses for customers.
* Customers could become frustrated.
* Maintenance required.
* They aren’t human ultimately so cannot be reliable in case of emergencies.
* Complex chatbots could cost more.
* Not all business can use chatbots.

# 8.Applications

* Customer Support
* **Suggest products**
* **Offer discounts**
* Fashion Assistant
* Smart Trip Assistant
* ​Healthcare Assistant

# 9. Future Scope

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. Conclusion

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.

# 11. Bibliography

1. https://[www.ibm.com/cloud/architecture/tutorials/cognitive\_discovery](http://www.ibm.com/cloud/architecture/tutorials/cognitive_discovery)
2. <https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started>
3. <https://www.wordstream.com/blog/ws/2017/10/04/chatbots>
4. <https://chatbotsmagazine.com/3-practical-applications-of-chatbots-for-customer-service-2bb7073bd2ac>
5. https://developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chatbot-on-nodered/
6. <http://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red>
7. <https://github.com/IBM/watson-discovery-sdu-with-assistant>

# Appendix

## Source Code

Node-Red Flow Source Code:

[{"id":"3879e5d1.87f75a","type":"tab","label":"Flow 1","disabled":false,"info":""},{"id":"a135d2c5.7ff1f","type":"ui\_form","z":"3879e5d1.87f75a","name":"","label":"","group":"8acdfea5.d95c6","order":2,"width":10,"height":2,"options":[{"label":"Enter the query","value":"input","type":"text","required":true,"rows":null}],"formValue":{"input":""},"payload":"","submit":"submit","cancel":"cancel","topic":"","x":70,"y":480,"wires":[["e2e03673.2ffd28"]]},{"id":"e2e03673.2ffd28","type":"function","z":"3879e5d1.87f75a","name":"input","func":"msg.payload=msg.payload.input;\nreturn msg;","outputs":1,"noerr":0,"x":210,"y":420,"wires":[["b31e791c.d14728","2408eaee.1b3b96"]]},{"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"]]},{"id":"1a31d881.e44577","type":"debug","z":"3879e5d1.87f75a","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","x":870,"y":180,"wires":[]},{"id":"881e2391.d752d","type":"ui\_text","z":"3879e5d1.87f75a","group":"3af1cd5f.b512d2","order":1,"width":12,"height":7,"name":"","label":"","format":"{{msg.payload}}","layout":"col-center","x":830,"y":540,"wires":[]},{"id":"b31e791c.d14728","type":"watson-conversation-v1","z":"3879e5d1.87f75a","name":"Customer Assistant","workspaceid":"d859c76b-6c40-43e0-84d1-41c10939137a","multiuser":false,"context":true,"empty-payload":false,"service-endpoint":"","timeout":"","optout-learning":false,"x":470,"y":320,"wires":[["f3891dc0.4d097"]]},{"id":"81285806.eed068","type":"inject","z":"3879e5d1.87f75a","name":"","topic":"","payload":"hi","payloadType":"str","repeat":"","crontab":"","once":false,"onceDelay":0.1,"x":90,"y":260,"wires":[["b31e791c.d14728"]]},{"id":"2408eaee.1b3b96","type":"ui\_text","z":"3879e5d1.87f75a","group":"8acdfea5.d95c6","order":6,"width":6,"height":2,"name":"","label":"You :","format":"{{msg.payload}}","layout":"row-spread","x":400,"y":540,"wires":[]},{"id":"8acdfea5.d95c6","type":"ui\_group","z":"","name":"Input","tab":"ae71c34b.31c07","order":1,"disp":true,"width":10,"collapse":false},{"id":"3af1cd5f.b512d2","type":"ui\_group","z":"","name":"Output","tab":"ae71c34b.31c07","order":2,"disp":true,"width":"12","collapse":false},{"id":"ae71c34b.31c07","type":"ui\_tab","z":"","name":"Home","icon":"dashboard","disabled":false,"hidden":false}]

Cloud Function Source Code:

|  |
| --- |
| /\*\* |
|  | \* |
|  | \* @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' |
|  | }); |