

# **PROJECT REPORT**

**KICKOFF DATE** - 5 May, 2020

## **DEVELOPMENT ENVIRONMENT -**

*GitHub Account* - <https://github.com/NishitaChandra>

*IBM Academic Initiative Account* - [nishitadinesh.chandra2017@vitstudent.ac.in](mailto:nishitadinesh.chandra2017@vitstudent.ac.in)

*IBM Cloud Account* - [nishitadinesh.chandra2017@vitstudent.ac.in](mailto:nishitadinesh.chandra2017@vitstudent.ac.in)

**Intelligent Customer Help Desk with Smart Document Understanding**

## **CONTENTS**

1. Introduction
2. About the project
3. Existing system
4. Proposed system
5. Flowchart of the model
6. Designing services used
7. Node-RED Starter Application
8. Watson discovery & services
9. IBM cloud functions
10. Final project
11. References

## **INTRODUCTION**

Watson is an IBM supercomputer that combines artificial intelligence (AI) and sophisticated analytical software for optimal performance as a "question answering" machine. The supercomputer is named for IBM's founder, Thomas J. Watson. The Watson supercomputer processes at a rate of 80 teraflops (trillion floating point operations per second). To replicate (or surpass) a high-functioning human's ability to answer questions, Watson accesses 90 servers with a combined data store of over 200 million pages of information, which it processes against six million logic rules. The system and its data are self-contained in a space that could accommodate 10 refrigerators.

Applications for Watson's underlying cognitive computing technology are almost endless. Because the device can perform text mining and complex analytics on huge volumes of unstructured data, it can support a search engine or an expert system with capabilities far superior to any previously existing.

## **ABOUT THE PROJECT**

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.

## **EXISTING SYSTEM**

At first, Chatbot can look like a normal app. There is an application layer, a database and APIs to call external services. In a case of the chatbot, UI is replaced with chat interface. While Chatbots are easy to use for users, it adds complexity for the app to handle.

There is a general worry that the bot can't understand the intent of the customer. The bots are first trained with the actual data. Most companies that already have a chatbot must be having logs of conversations. Developers use that logs to analyze what customers are trying to ask and what does that mean. With a combination of Machine Learning models and tools built, developers match questions that customer asks and answers with the best suitable answer. For example: If a customer is asking "Where is my payment receipt?" and "I have not received a payment receipt", mean the same thing. Developers strength is in training the models so that the chatbot is able to connect both of those questions to correct intent and as an output produces the correct answer. If there is no extensive data available, different APIs data can be used to train the chatbot.

## **HOW CHATBOTS ACTUALLY WORK?**

The chatbots work by adopting 3 classification methods:

### **Pattern Matchers:**

Bots use pattern matching to classify the text and produce a suitable response for the customers. A standard structure of these patterns is "Artificial Intelligence Markup Language" (AIML).

A simple pattern matching example:

```
<aiml version = "1.0.1" encoding = "UTF-8"?>
  <category>
    <pattern> WHO IS ABRAHAM LINCOLN </pattern>
    <template>Abraham Lincoln was the US President during American civil war.</template>
  </category>

  <category>
    <pattern>DO YOU KNOW WHO * IS</pattern>
    <template>
      <srai>WHO IS <star/></srai>
    </template>
  </category>
</aiml>
```

The machine then gives and output:

**Human:** Do you know who Abraham Lincoln is?

**Robot:** Abraham Lincoln was the US President during American civil war.

### Algorithms:

For each kind of question, a unique pattern must be available in the database to provide a suitable response. With lots of combination on patterns, it creates a hierarchical

structure. We use algorithms to reduce the classifiers and generate the more manageable structure. Computer scientists call it a "Reductionist" approach- in order to give a simplified solution, it reduces the problem.

Multinational Naive Bayes is the classic algorithm for text classification and NLP. For an instance, let's assume a set of sentences are given which are belonging to a particular class. With new input sentence, each word is counted for its occurrence and is accounted for its commonality and each class is assigned a score. The highest scored class is the most likely to be associated with the input sentence.

For example Sample Training set

class: greeting  
"How you doing?"  
"good morning"  
"hi there"

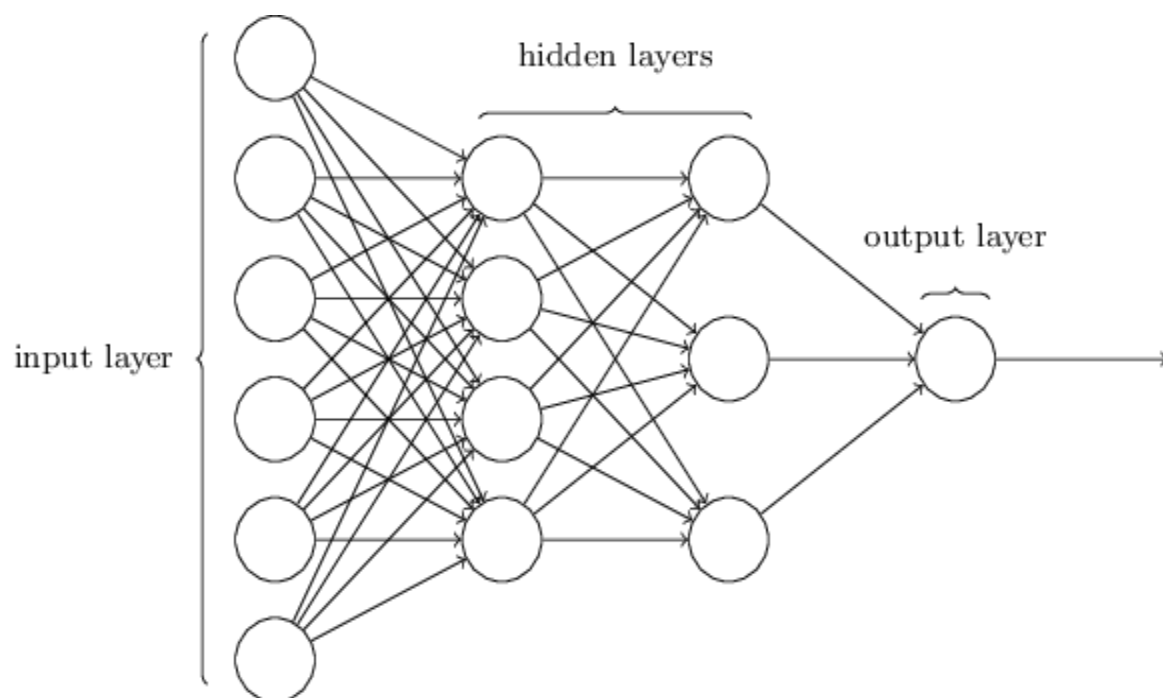
Few sample Input sentence classification:

input: "Hello good morning"  
term: "hello" (no matches)  
Term: "good" (class: greeting)  
term: "morning" (class: greeting)  
classification: greeting (score=2)

With the help of equation, word matches are found for given some sample sentences for each class. Classification score identifies the class with the highest term matches but it also has some limitations. The score signifies which intent is most likely to the sentence but does not guarantee it is the perfect match. Highest score only provides the relativity base.

### **Artificial Neural Networks:**

Neural Networks are a way of calculating the output from the input using weighted connections which are calculated from repeated iterations while training the data. Each step through the training data amends the weights resulting in the output with accuracy.



As discussed earlier here also, each sentence is broken down into different words and each word then is used as input for the neural networks. The weighted connections are then calculated by different iterations through the training data thousands of times. Each time improving the weights to making it accurate. The trained data of neural network is a comparable algorithm more and less code. When there is a comparably small sample, where the training sentences have 200 different words and 20 classes,



then that would be a matrix of  $200 \times 20$ . But this matrix size increases by  $n$  times more gradually and can cause a huge number of errors. In this kind of situations, processing speed should be considerably high.

There are multiple variations in neural networks, algorithms as well as patterns matching code. Complexity may also increase in some of the variations. But the fundamental remains the same, and the important work is that of classification.

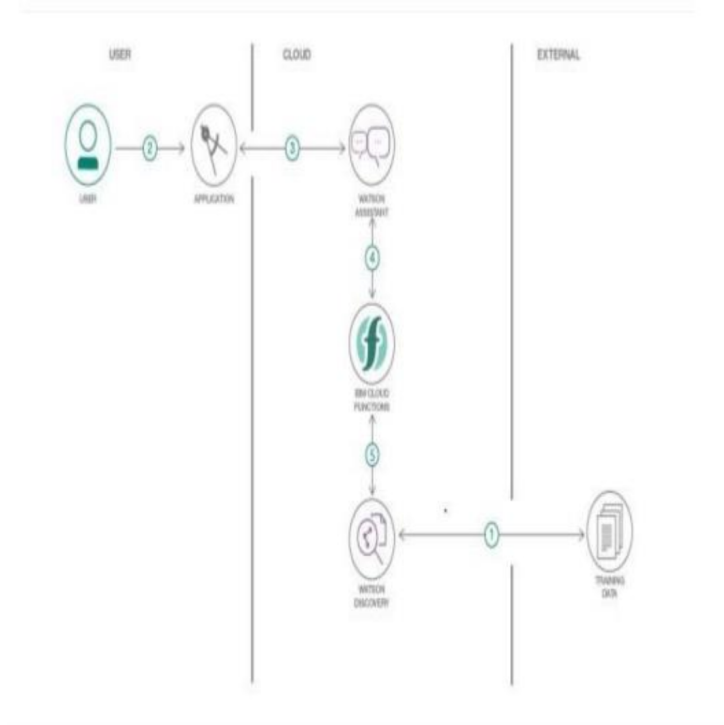
### **PROPOSED SYSTEM**

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.

We are using Hukseflux Thermal sensors HTR01 user manual. At the end of the project Watson will be trained with the various user's questions and answer them specifically.

## FLOWCHART OF MODEL



## DESIGNING SERVICES USED

1. IBM Cloud
2. IBM Watson
3. Node-RED
4. Python
5. Watson Assitance

## NODE-RED STARTER APPLICATION

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.

1. Log in to IBM Cloud.
2. Open the catalog and search for **node-red**.
3. Click on the **Software** tab.
4. Click on the **Node-RED App** tile.
5. This will show you an overview of the Starter Kit and what it provides.
6. Click on the **Create app** button to continue.

Step 2. Configure your application

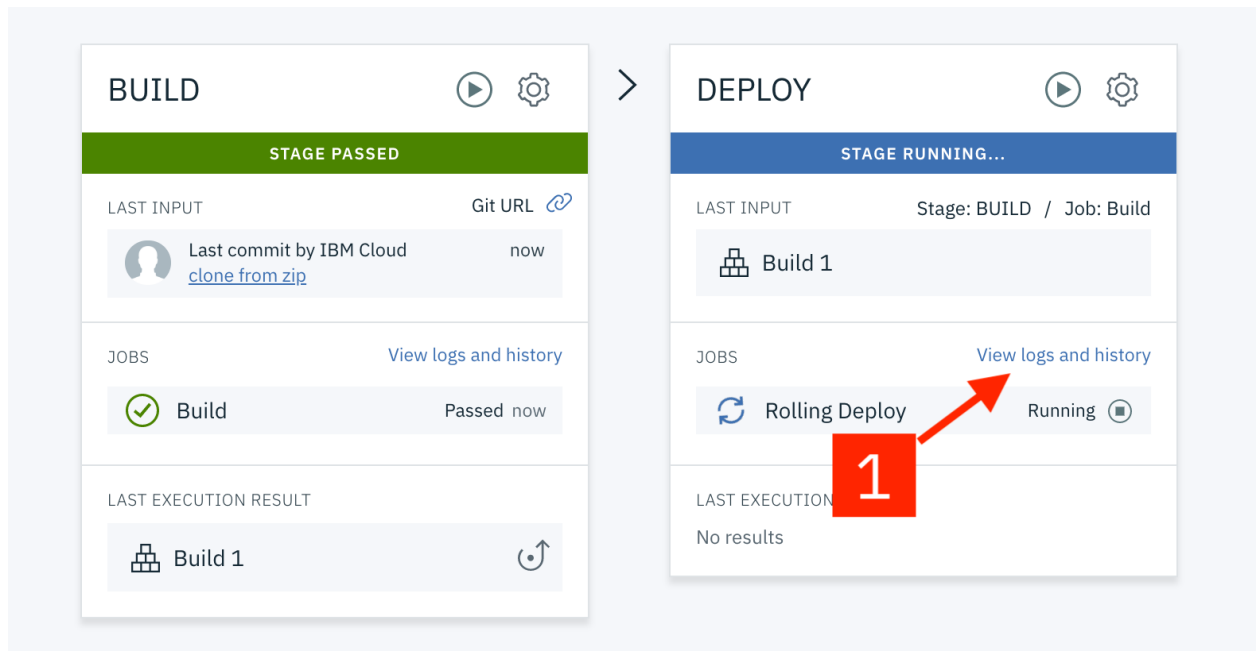
Now you need to configure the Node-RED Starter application.

1. 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.
2. 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.
3. 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.

1. On the next screen, click the **Deploy your app** button to enable the *Continuous Delivery* feature for your application.
2. 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.
3. 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.
4. 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.
5. Select the **region** to create the **DevOps toolchain**.
6. Click **Create**. This will take you back to the application details page.
7. 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.
8. Click on the **In progress** link to see the full status of the Delivery Pipeline.



9.

10. 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.

The final screen shown is this

The screenshot shows the IBM Cloud console interface for a service named "Node RED LTI V1". The top navigation bar includes "IBM Cloud", a search bar, and links to "Catalog", "Docs", "Support", "Manage", and a user profile "Nishita Chandr...". The breadcrumb trail is "Resource list / App details /".

**Details**

- App URL: *You must deploy your app first*
- Source: <https://us-south.git.cloud.ibm.com/nishitadinesh.chandra2017/NodeREDLTI V1>
- Resource group: **Default**
- Deployment target: *You must deploy your app first*
- Created: 5/27/2020

**Services (1)** [Connect existing services](#) [Create service](#)

Name	Resources	Actions
<b>Cloudant</b>	<a href="#">Documentation</a>	

**Deployment Automation**

- Name: **NodeREDLTI V1**
- Location: **Dallas**
- Tool integrations:

**Delivery Pipelines**

- Name: [NodeREDLTI V1](#)
- Status: [Failure](#)
- Last input: Last commit by IBM Cloud (2 hours ago)
- [Clone from zip](#)

**Credentials** [Show](#)

```
{
  "cloudant": {
    "name": "node-red-lti-v1-cloudant-1590556059583",
    "credentials": {
      "apikey": "*****",
      "host": "254bda70-ae98-4724-bbcc-554cab7590eb-bluemix.clo"
    }
  }
}
```

## WATSON ASSISTANT

1. In IBM cloud go to Catalog, search Watson Assistant.
2. Let the details be as default values and click Create.
3. Let the services be activated.

The screenshot shows the IBM Cloud console interface for a service named "Watson Assistant-09". The top navigation bar includes "IBM Cloud", a search bar, and links to "Catalog", "Docs", "Support", "Manage", and a user profile "Nishita Chandr...". The breadcrumb trail is "Resource list /".

**Watson Assistant-09** ● Active [Add tags](#)

**Details** [Actions...](#)

**Manage**

- Service credentials
- Plan
- Connections

**Start by launching the tool**

[Launch Watson Assistant](#) [Getting started tutorial](#) [API reference](#)

**Credentials** [Download](#) [Show credentials](#)

API key:

URL:

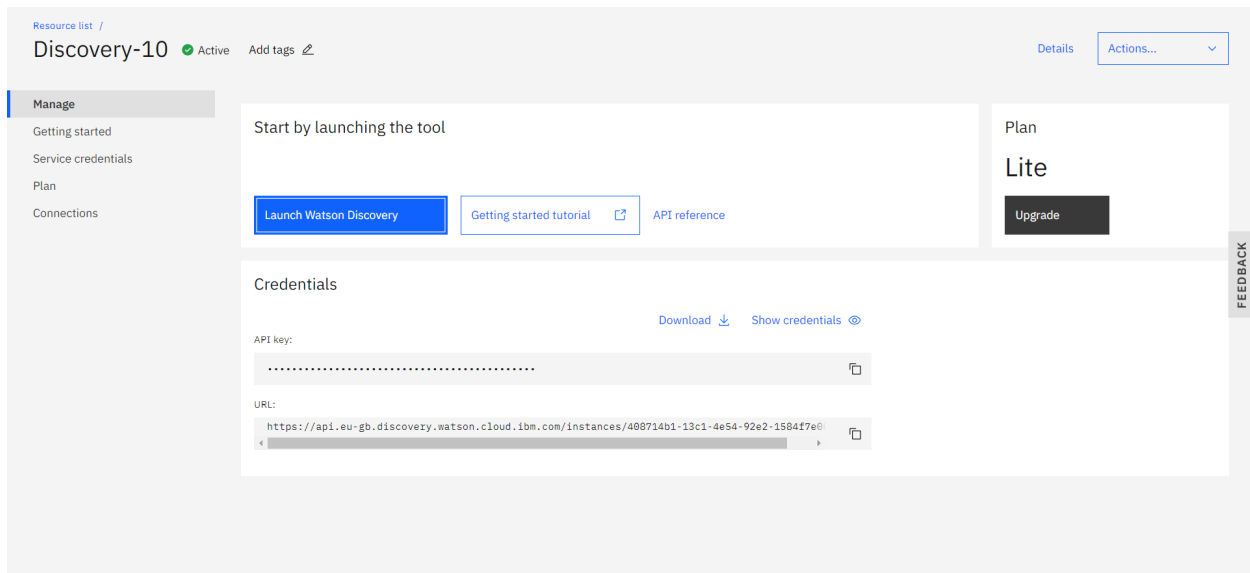
**Plan**

**Lite**

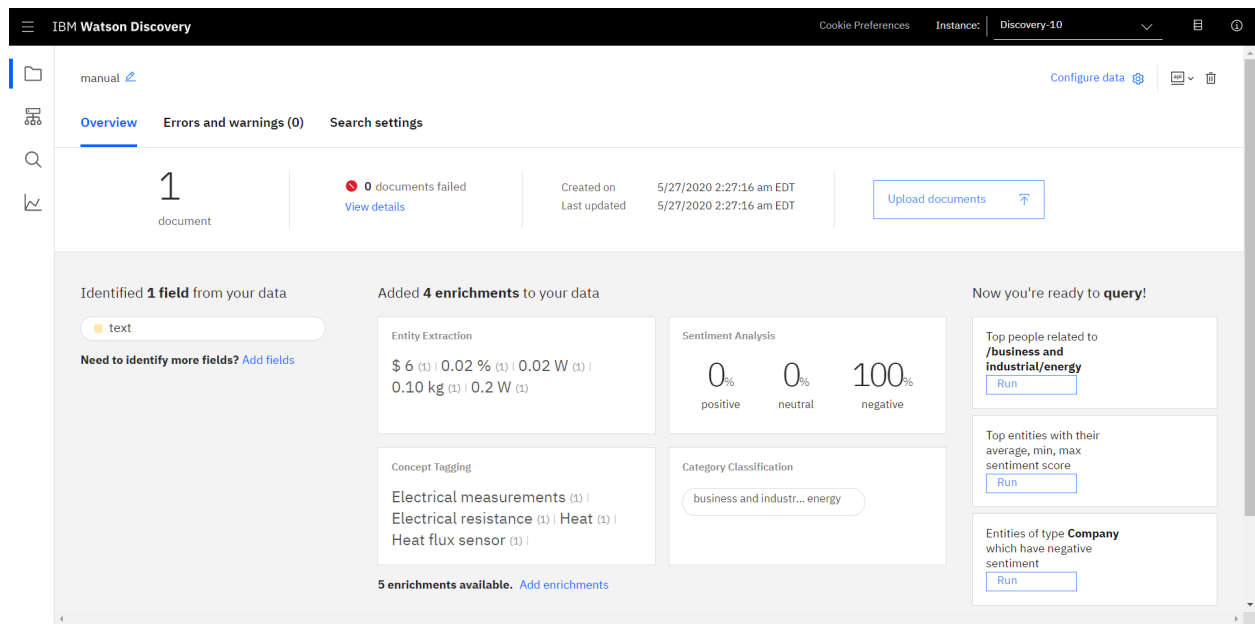
[Upgrade](#)

## WATSON DISCOVERY

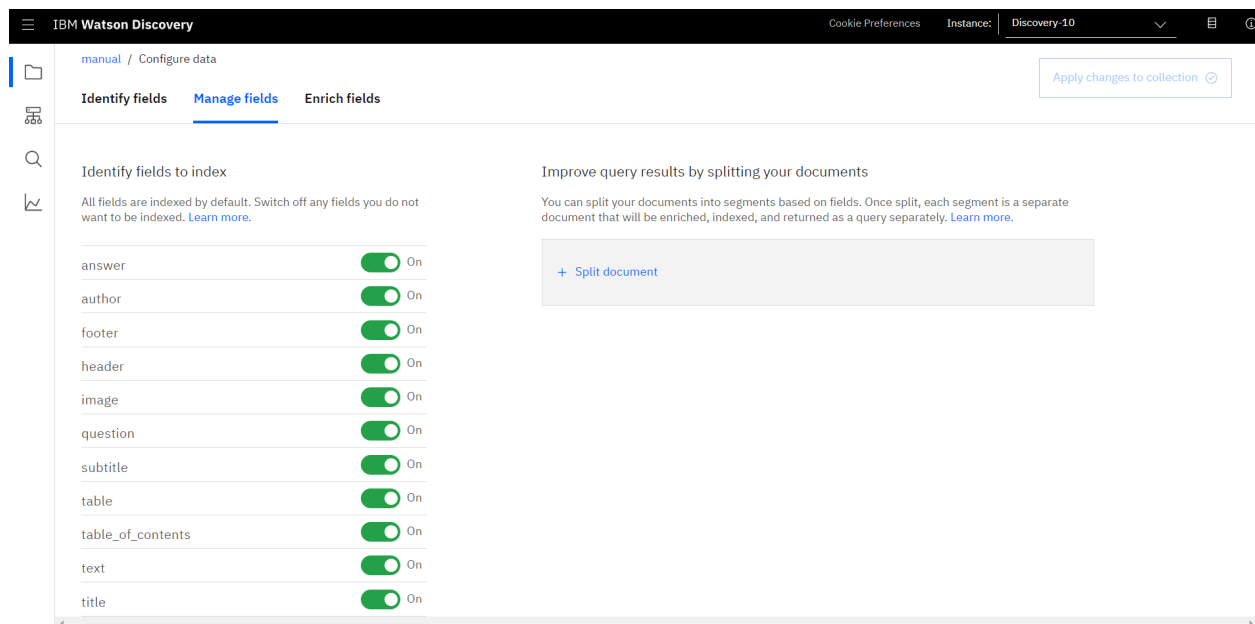
1. Go to Catalog, search Watson Discovery.
2. Let the details be as default values and click Create.
3. Let the services be activated.



4. Click on Launch Watson Discovery.
5. Click Upload your own data, give a name to the collection.
6. In the next page, click on upload and select the user manual you want Watson to be trained at, here we use a heater manual.
7. Configure the data.

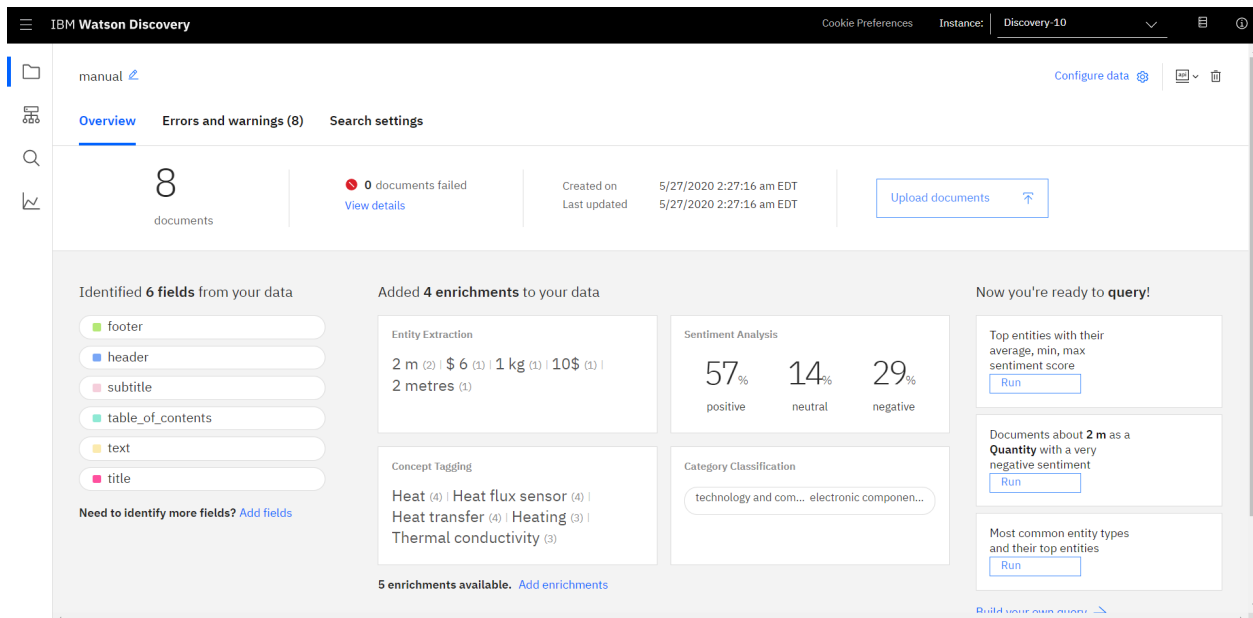


## 8. Manage fields of the data, train Watson to recognize text, questions, subtitles, table of contents.



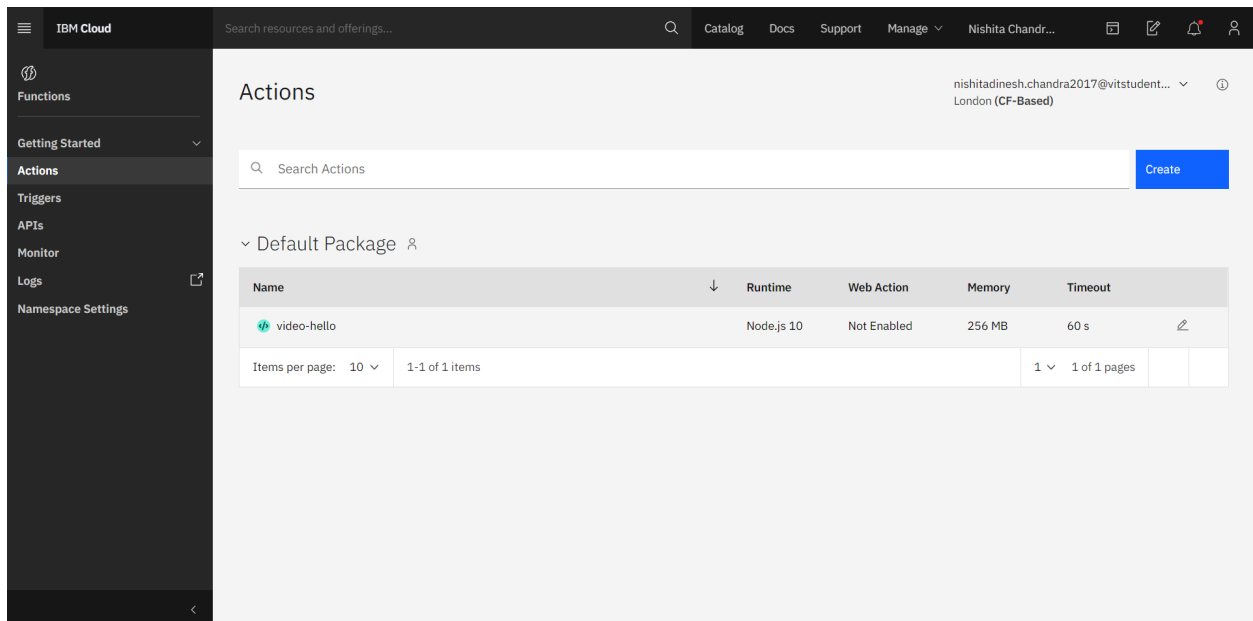
The final page looks like





## IBM CLOUD FUNCTIONS

1. Search functions in the search box.
2. In IBM functions go to Actions.
3. Create a new Action with name video-hello.



4. Type the code.

5. Change,add the parameters according to the watson api key etc.

The screenshot shows the IBM Cloud Functions console. The top navigation bar includes the IBM Cloud logo, a search bar, and links to Catalog, Docs, Support, and Manage. The user's name, Nishita Chandr..., is displayed. The breadcrumb trail indicates the current location: Functions / Actions / video-hello. The function name 'video-hello' is prominently displayed, along with a 'Web Action' icon. The namespace is noted as nishitadinesh.chandra2017@vitsstudent.ac.in\_dev(London). On the left sidebar, the 'Parameters' tab is selected, showing a list of parameters for the function. The parameters are as follows:

Parameter Name	Parameter Value
url	"https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/408714b1-13c1-4e5"
environment_id	"b28f8eb0-9cf3-4ec9-b07d-14ef9a2aaa6f"
collection_id	"14e553ba-3e4b-4179-88b2-e7352e523aa9"
iam_apikey	"HL5NFGXtHacp-EgSbqdGF2uSEkUoh_Kvu1e4OnaLmWS_"

An 'Add Parameter' link is visible in the top right corner of the parameters section.

6. Enable the Web Action and copy the URL.

7. Launch the Watson Assistant.

8. Click on create Assistant.

9. Create skills for the Assistant.

IBM Watson Assistant Lite

Upgrade

Customer Care Sample Skill

Save new version

Try it

Intents

Entities

Dialog

Options

Analytics

Versions

Content Catalog

Intents (10) ↑

#Cancel

Cancel the current request

5 minutes ago

7

#Customer\_Care\_Appointments

Schedule or manage an in-store appointment.

5 minutes ago

20

#Customer\_Care\_Store\_Hours

Find business hours.

5 minutes ago

48

#Customer\_Care\_Store\_Location

Locate a physical store location or an address.

5 minutes ago

25

#General\_Connect\_to\_Agent

Request a human agent.

5 minutes ago

47

#General\_Greetings

Greetings

5 minutes ago

30

#Goodbye

Good byes

5 minutes ago

6

#Help

Ask for help

5 minutes ago

8

#Product\_Information

a few seconds ago

5

Showing 1–10 of 10 intents

1

1 of 1 pages

Create intent +

10. Edit the dialog to give out better answers.

IBM Watson Assistant Lite

Upgrade

Customer Care Sample Skill

Save new version

Intents

Entities

Dialog

Options

Analytics

Versions

Content Catalog

Ask about product

Node name will be shown to customers for disambiguation so use something descriptive.

Settings

If assistant recognizes

#Product\_Information

+

Intents

#Product\_Information

How to turn on heater

Text

Enter response text

Response variations are set to **sequential**. Set to [random](#) | [multiline](#)

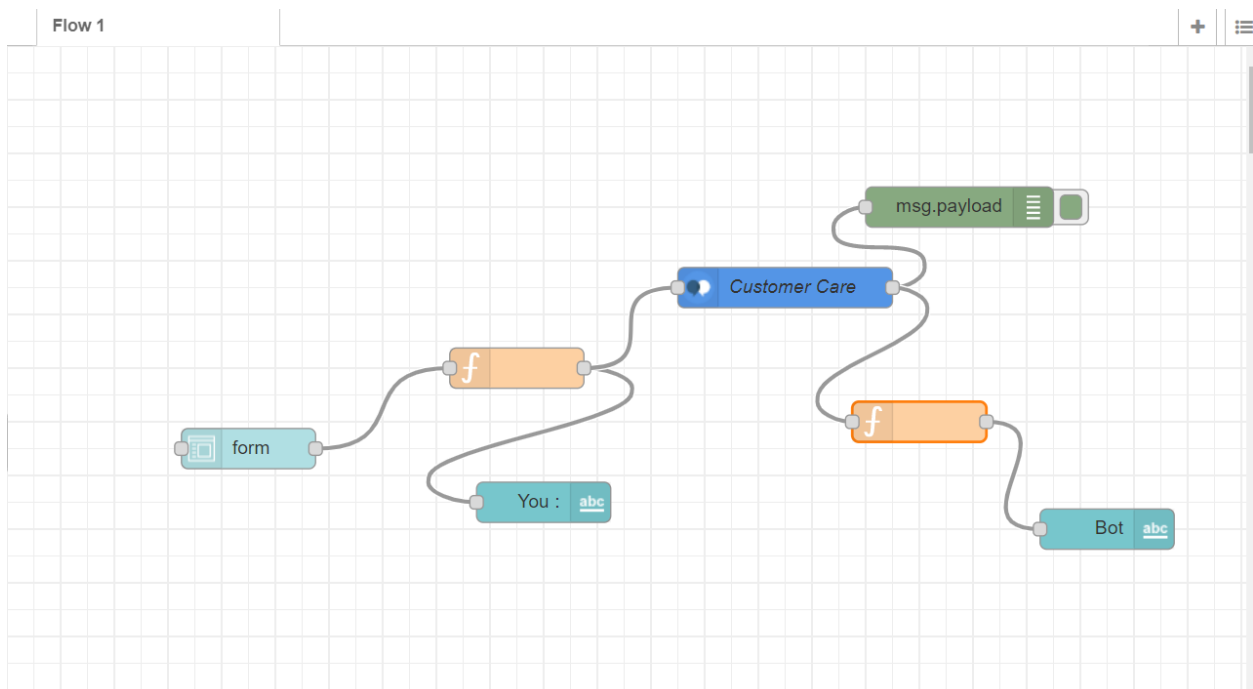
Learn more

Add response type +

11. Add the URL generated in cloud to the Webhooks.
12. Go to the Node-RED editor and create your own flow.
13. Install the node-red dashboard palette.



14. Create nodes and edit them.

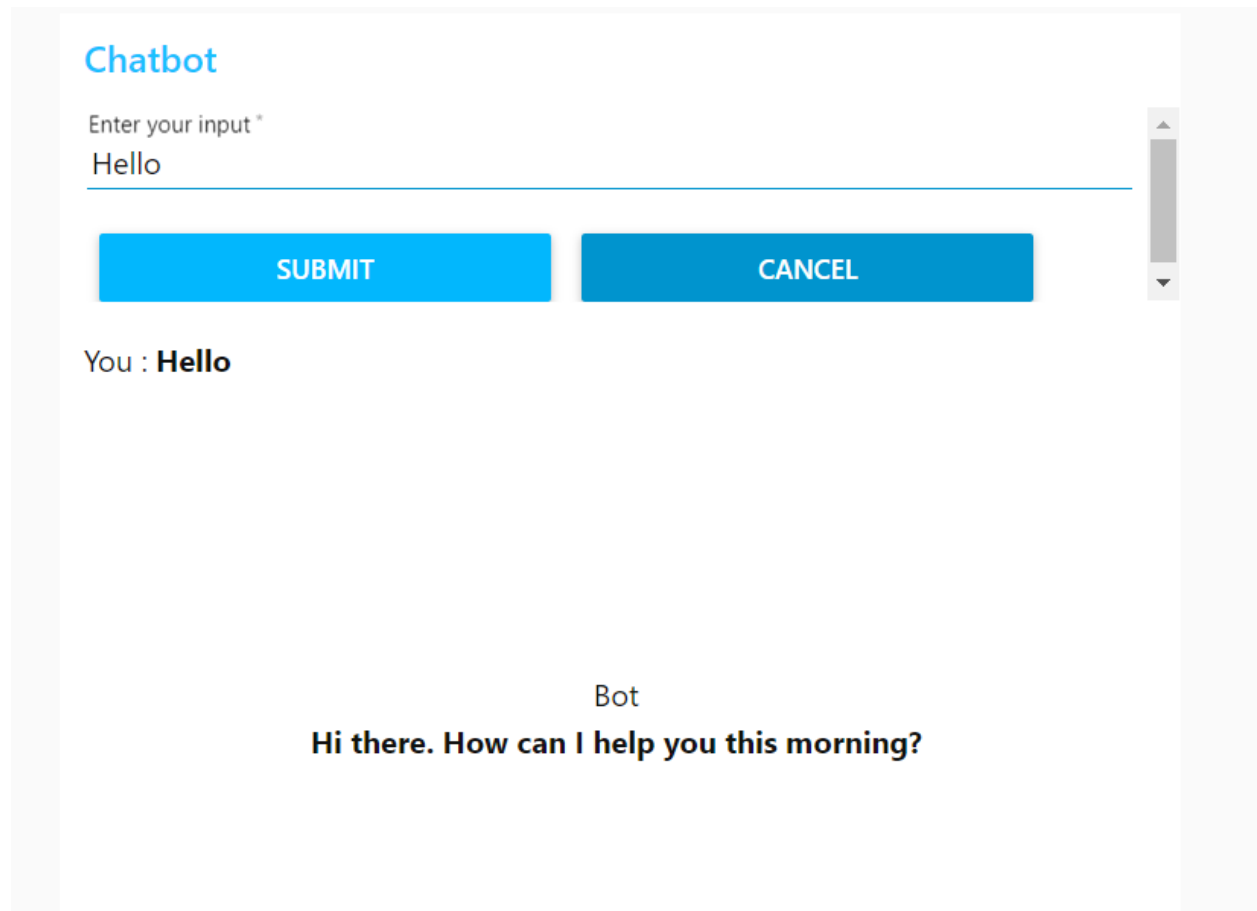


15. Edit the forms by clicking on them. Once done click Deploy.

16. Take the dashboard URL and add /ui to open the chatbot.

## FINAL PROJECT

Here are some screenshots of the final project prepared.



**Chatbot**

Enter your input \*

Hello

**SUBMIT** **CANCEL**

You : **Hello**

Bot

**Hi there. How can I help you this morning?**

## Chatbot

Enter your input \*

How to turn on heater

SUBMIT

CANCEL

You : **How to turn on heater**

Bot

**"Please refer to section 2.1 for more details on how to compute the heater power and applied heat flux. Suggested HTR01 wiring is shown in figure 4.2.1. The heater serial number and resistance are shown on the HTR01 product certificate and on the sticker on the strain relief block."**

## Chatbot

Enter your input \*

How to change time of heater

SUBMIT

CANCEL

You : **How to change time of heater**

Bot

**"Upon installing the heat flux sensor and HTR01, a reference measurement should be made. A time trace of the heater power, the measured heat flux and the measured sensor temperature should be stored as reference data."**

## Chatbot

Enter your input \*

How to set time in heater

SUBMIT

CANCEL

You : **How to set time in heater**

Bot

**"This can be done accurately by using the four heater wires to conduct a four-point resistance measurement. 3. Record a time trace of the heater power, the measured heat flux and the sensor temperature; the same parameters as in the reference data. Normalise the data by the heater power."**



## Chatbot

Enter your input \*

What is a heater

SUBMIT

CANCEL

You : **What is a heater**

Bot

**"If a variable heat flux is required, the heater can either be connected via a solid state relay controlled by a pulse-width modulated (PWM) signal or to a programable power supply (see figure 4.2.1). The HTR01 electrical connections are explained in table 4.2.1 When connecting HTR01, always observe the rated heater voltage."**

## REFERENCES

1. <https://www.ibm.com/cloud/get-started>
2. <https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/>
3. <https://github.com/watson-developer-cloud/node-red-labs>
4. [https://www.w3schools.com/howto/howto\\_make\\_a\\_website.asp](https://www.w3schools.com/howto/howto_make_a_website.asp)
5. <https://www.ibm.com/watson/products-services>
6. <https://developer.ibm.com/components/watson-assistant/series/learning-path-watson-assistant>
7. <https://developer.ibm.com/articles/introduction-watson-discovery/>