PROJECT REPORT

Internship by smartinternz

TOPIC: Intelligent Customer Help Desk with Smart Document Understanding

NAME: Sruthi Prabakaran P

DOMAIN: Machine Learning

EMAIL: 1999sruthi@gmail.com

TABLE OF CONTENTS

1.INTRODUCTION

A **Chatbot** is a software application used to conduct an online chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent. Designed to convincingly simulate the way a human would behave as a conversational partner, chatbot systems typically require continuous tuning and testing, and many in production remain unable to adequately converse or pass the industry standard Turing test. The term "ChatterBot" was originally coined by Michael Mauldin (creator of the first Verbot) in 1994 to describe these conversational programs.

Chatbots are typically used in dialog systems for various purposes including customer service, request routing, or for information gathering. While some chatbot applications use extensive word-classification processes, Natural Language processors, and sophisticated AI, others simply scan for general keywords and generate responses using common phrases obtained from an associated library or database.

Today, most chatbots are accessed on-line via website popups, or through virtual assistants such as Google Assistant, Amazon Alexa, or messaging apps such as Facebook Messenger or WeChat. Chatbots are typically classified into usage categories that include: commerce (e-commerce via chat), education, entertainment, finance, health, news, and productivity.

Driven by AI, automated rules, natural-language processing (NLP), and machine learning (ML), chatbots process data to deliver responses to requests of all kinds.

There are two main types of chatbots.

- **Task-oriented (declarative) chatbots** are single-purpose programs that focus on performing one function. Using rules, NLP, and very little ML, they

generate automated but conversational responses to user inquiries. Interactions with these chatbots are highly specific and structured and are most applicable to support and service functions—think robust interactive FAQs. Task-oriented chatbots can handle common questions, such as queries about hours of business or simple transactions that don't involve a variety of variables. Though they do use NLP so end users can experience them in a conversational way, their capabilities are fairly basic. These are currently the most commonly used chatbots.

- **Data-driven and predictive (conversational) chatbots** are often referred to as virtual assistants or digital assistants, and they are much more sophisticated, interactive, and personalized than task-oriented chatbots. These chatbots are contextually aware and leverage natural-language understanding (NLU), NLP, and ML to learn as they go. They apply predictive intelligence and analytics to enable personalization based on user profiles and past user behavior. Digital assistants can learn a user's preferences over time, provide recommendations, and even anticipate needs. In addition to monitoring data and intent, they can initiate conversations. Apple's Siri and Amazon's Alexa are examples of consumer-oriented, data-driven, predictive chatbots.

2.LITERATURE SURVEY

The origin of the chatbot arguably lies with Alan Turing's 1950s vision of intelligent machines. Artificial intelligence, the foundation for chatbots, has progressed since that time to include superintelligent supercomputers such as IBM Watson.

In 1950, Alan Turing's famous article "Computing Machinery and Intelligence" was published, which proposed what is now called the Turing test as a criterion of intelligence. This criterion depends on the ability of a computer program to

impersonate a human in a real-time written conversation with a human judge to the extent that the judge is unable to distinguish reliably—on the basis of the conversational content alone—between the program and a real human. The notoriety of Turing's proposed test stimulated great interest in Joseph Weizenbaum's program ELIZA, published in 1966, which seemed to be able to fool users into believing that they were conversing with a real human. However Weizenbaum himself did not claim that ELIZA was genuinely intelligent, and the introduction to his paper presented it more as a debunking exercise:

In artificial intelligence ... machines are made to behave in wondrous ways, often sufficient to dazzle even the most experienced observer. But once a particular program is unmasked, once its inner workings are explained ... its magic crumbles away; it stands revealed as a mere collection of procedures ... The observer says to himself "I could have written that". With that thought, he moves the program in question from the shelf marked "intelligent", to that reserved for curios ... The object of this paper is to cause just such a re-evaluation of the program about to be "explained". Few programs ever needed it more.

ELIZA's key method of operation (copied by chatbot designers ever since) involves the recognition of clue words or phrases in the input, and the output of the corresponding pre-prepared or pre-programmed responses that can move the conversation forward in an apparently meaningful way (e.g. by responding to any input that contains the word 'MOTHER' with 'TELL ME MORE ABOUT YOUR FAMILY'). Thus an illusion of understanding is generated, even though the processing involved has been merely superficial. ELIZA showed that such an illusion is surprisingly easy to generate because human judges are so ready to give the benefit of the doubt when conversational responses are *capable of being interpreted* as "intelligent".

Interface designers have come to appreciate that humans' readiness to interpret computer output as genuinely conversational—even when it is actually based on

rather simple pattern-matching—can be exploited for useful purposes. Most people prefer to engage with programs that are human-like, and this gives chatbot-style techniques a potentially useful role in interactive systems that need to elicit information from users, as long as that information is relatively straightforward and falls into predictable categories. Thus, for example, online help systems can usefully employ chatbot techniques to identify the area of help that users require, potentially providing a "friendlier" interface than a more formal search or menu system. This sort of usage holds the prospect of moving chatbot technology from Weizenbaum's "shelf ... reserved for curios" to that marked "genuinely useful computational methods".

The original chatbot was the phone tree, which led phone-in customers on an often cumbersome and frustrating path of selecting one option after another to wind their way through an automated customer service model. Enhancements in technology and the growing sophistication of AI, ML, and NLP evolved this model into pop-up, live, onscreen chats. And the evolutionary journey has continued.

With today’s digital assistants, businesses can scale AI to provide much more convenient and effective interactions between companies and customers—directly from customers’ digital devices.

3.PROJECT DESCRIPTION

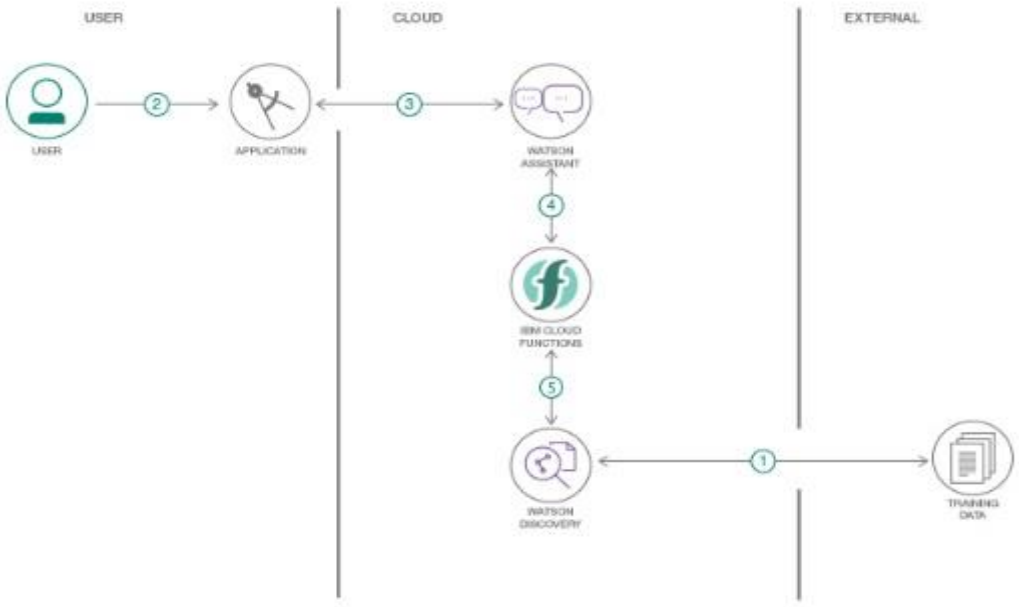
The typical customer care chatbot can answer simple questions, such as store locations and hours, directions, and maybe even making appointments. When a question falls outside of the scope of the pre-determined question set, the option is typically to tell the customer the question isn’t valid or offer 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 owner manual. So now, instead of "Would you like to speak to a customer representative?" we can return relevant sections of the owner's 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 owner's manual is important and what is not. This will improve the answers returned from the queries.

With the current issue in mind, this chatbot is about providing basic information about the novel coronavirus.

FLOW DIAGRAM:



- Firstly, Watson discovery annotates and gets trained on the document used to answer all the in-detail queries.

- When a user requests information from the bot, Watson assistant filters out only the required entities and searches for the information based on the entity.
- If relevant information is found in the assistant, it is returned.
- Whenever an appropriate answer is not found in the assistant, Watson discovery is initiated and the relevant information is returned to the node-red application.

3.1. PURPOSE

A usual chatbot answers only the basic queries by the user. All other detailed queries are connected to humans. But, the smart document understanding capability of Watson discovery along with Watson assistant and node-red complex queries could now be answered by the bot reducing human intervention.

3.2. EXISTING ISSUES:

The novel coronavirus is a worldwide pandemic. While the doctors and scientists are trying hard to connect information and arrive at a possible vaccine, the public is left with little but scattered information. There is no one place where people can receive all the right information, like a chatbot.

3.3 PROPOSED SYSTEM:

- The queries about the COVID-19 are posted in the chatbot
- Front end application is built using Node-Red
- Watson Assistant processes all the queries put forth by the user
- Queries that cannot be answered using the assistant and that requires human assistant are taken care of by Watson Discovery.
- Watson Assistant and Discovery are connected together using webhooks

3.4. PROJECT SCOPE

- 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

3.5. SYSTEM CONFIGURATION

1. Create IBM cloud services
2. Configure Watson discovery
3. Create Node-Red application (UI) and configure with Watson assistant
4. Create an action using IBM functions
5. Connect Watson assistant with a Discovery Instance
6. Deploy and run the application

4. SYSTEM IMPLEMENTATION

4.1. Create IBM Cloud Services

a. Watson Discovery
b. Watson Assistant
c. Node-Red application

4.2. Configure Watson Discovery

Upon creating Watson discovery service, launch the service instance created. The service instance can be accessed in the services part of the resources list page. Import the required document that needs to be learned. After some time the application is trained with the uploaded document. We can check the level of training by checking with a few queries in the build queries tab. We can configure the data using the configure data option. This lets us segregate the data based on various options like text, title, subtitle, table of contents, etc. After identifying the required fields, the next step is to manage fields by selecting the data which is required and splitting the document based on fields.

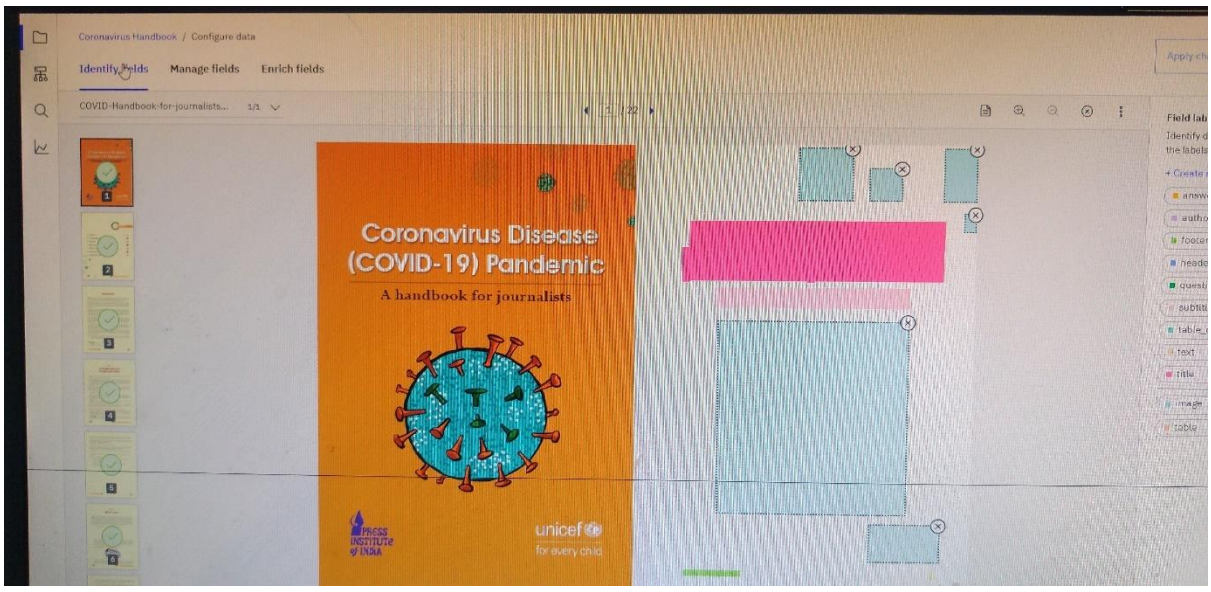


Figure 4.1. Data configuration- Identifying fields

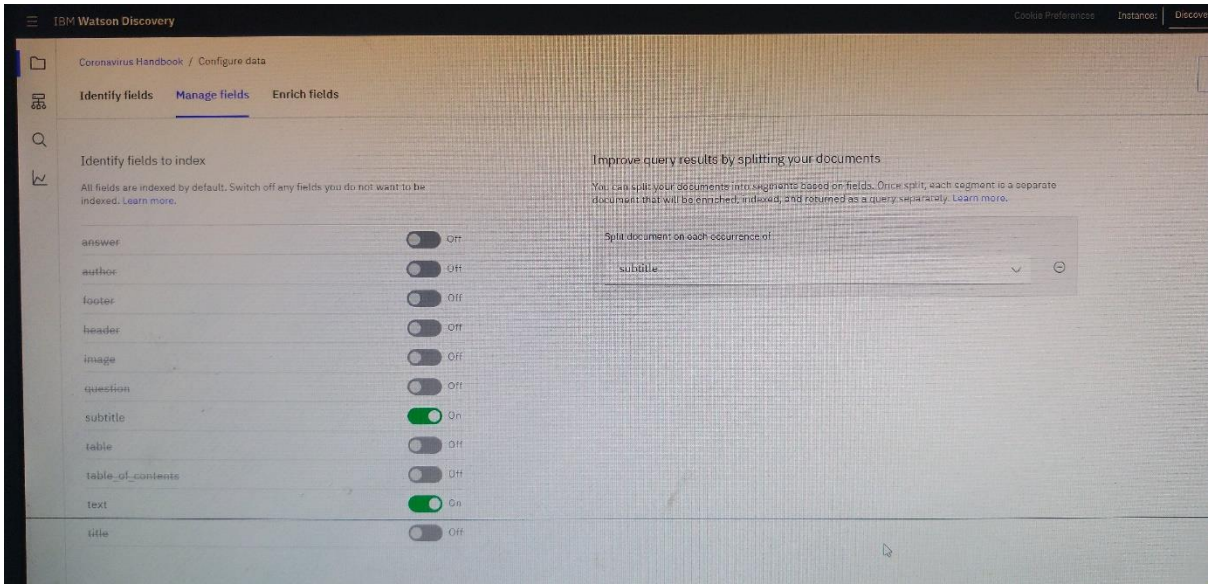


Figure 4.2. Managing Fields

4.2. Create Node-Red application

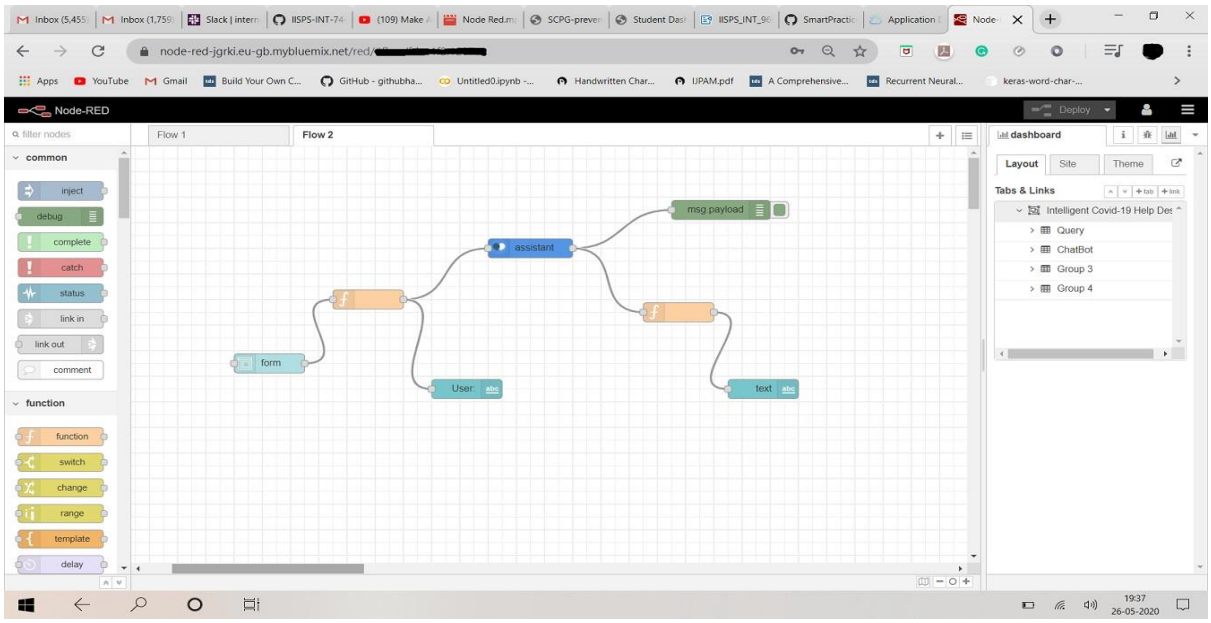


Figure 4.3. Node-Red application

Create a node-red instance in the node-red app of IBM Cloud. After creating an instance with relevant details, deploy the instance in order to enable continuous deliverable. Once it is done, select on the node-red service instance in the services

tab of the resources list page. Upon clicking on the visit URL link, the corresponding application is started. Here use the widgets to create a chatbot application.

4.3. Create an action using IBM Functions

IBM Functions are a distributed compute service that executes an application logic upon user requests from an app or a user. After creating a new action, write the corresponding code by using the appropriate parameters to connect to the Watson discovery instance. In our case, the purpose of the function is to connect to Watson discovery, validate the credentials, and return the resolved data from the query to the node-red application.

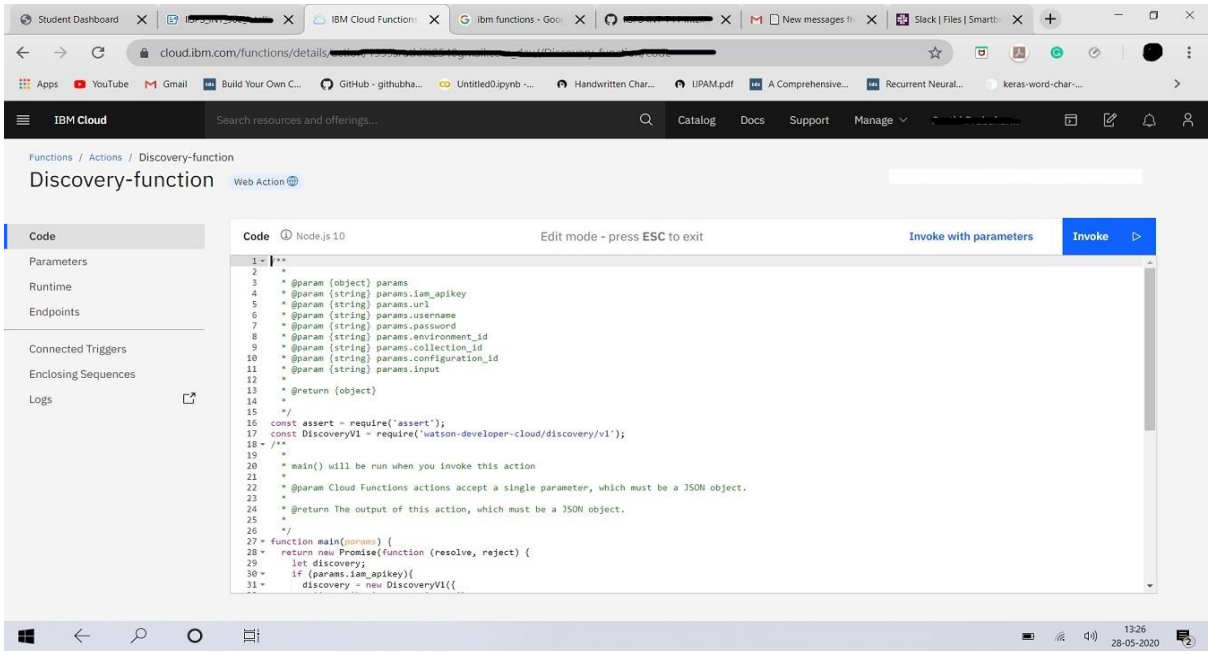


Figure 4.4. IBM Action

The parameters used in the action code are provided in the parameters tab. The same parameter name used in code must be provided in the parameters tab with credentials from the Watson discovery like API key, Environment ID, Collection ID, etc. Ensure to provide the parameter values in double-quotes.

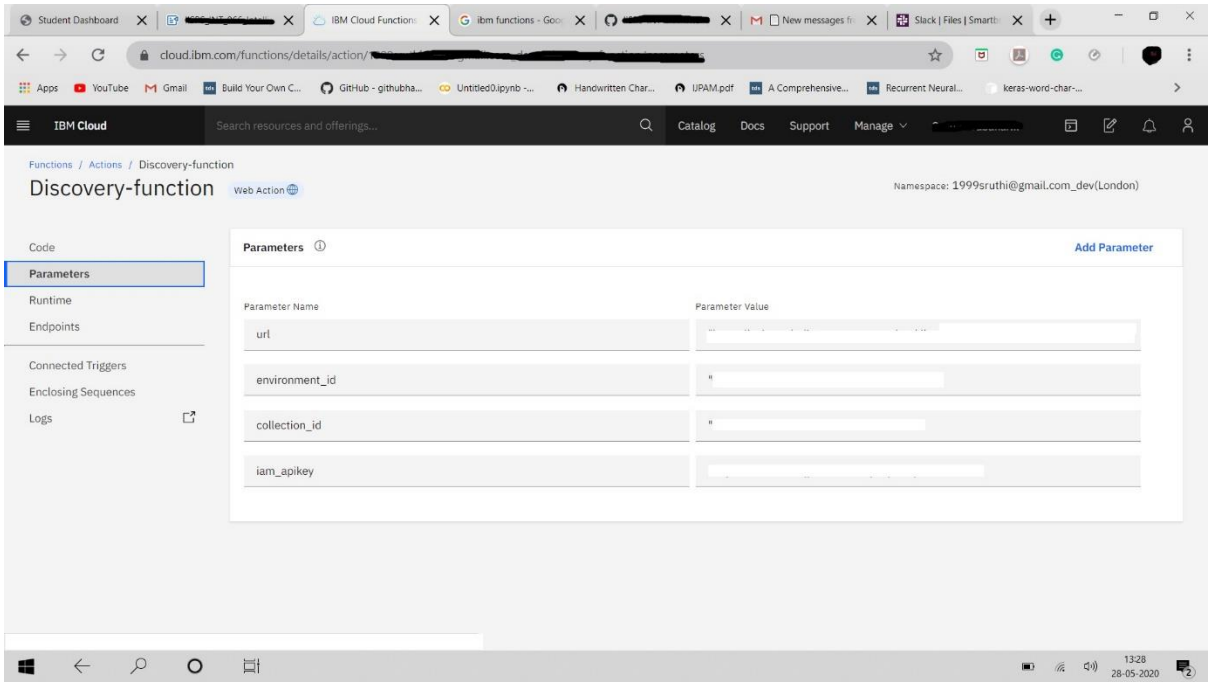


Figure 4.5. Action Parameters

4.4. Connect Watson Assistant with Watson Discovery

In the node-red application, provide the discovery credentials in the assistant node. Ensure to copy the correct credentials.

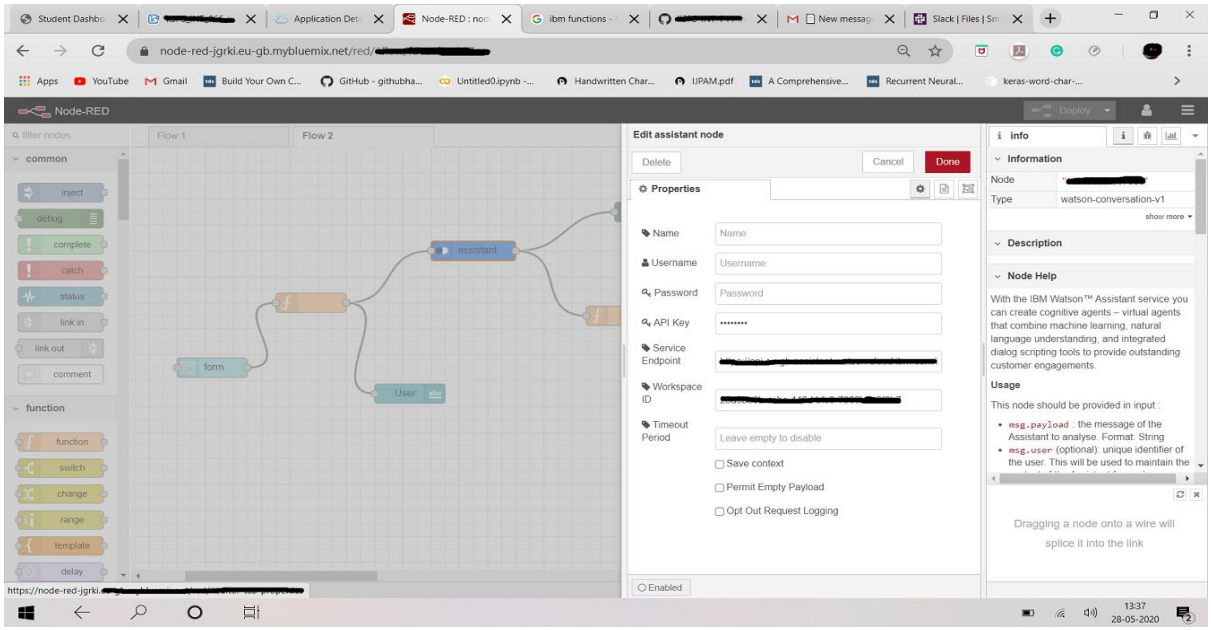


Figure 4.6. Assistant node configuration

5. PROJECT FLOW

1. Upload the coronavirus handbook from the internet(UNICEF)
2. Configure the Watson discovery instance with the handbook
3. Create Watson assistant service instance
4. Create a node-red application with front-end and back-end (assistant and discovery)
5. Test and deploy the application.

6. OUTPUT

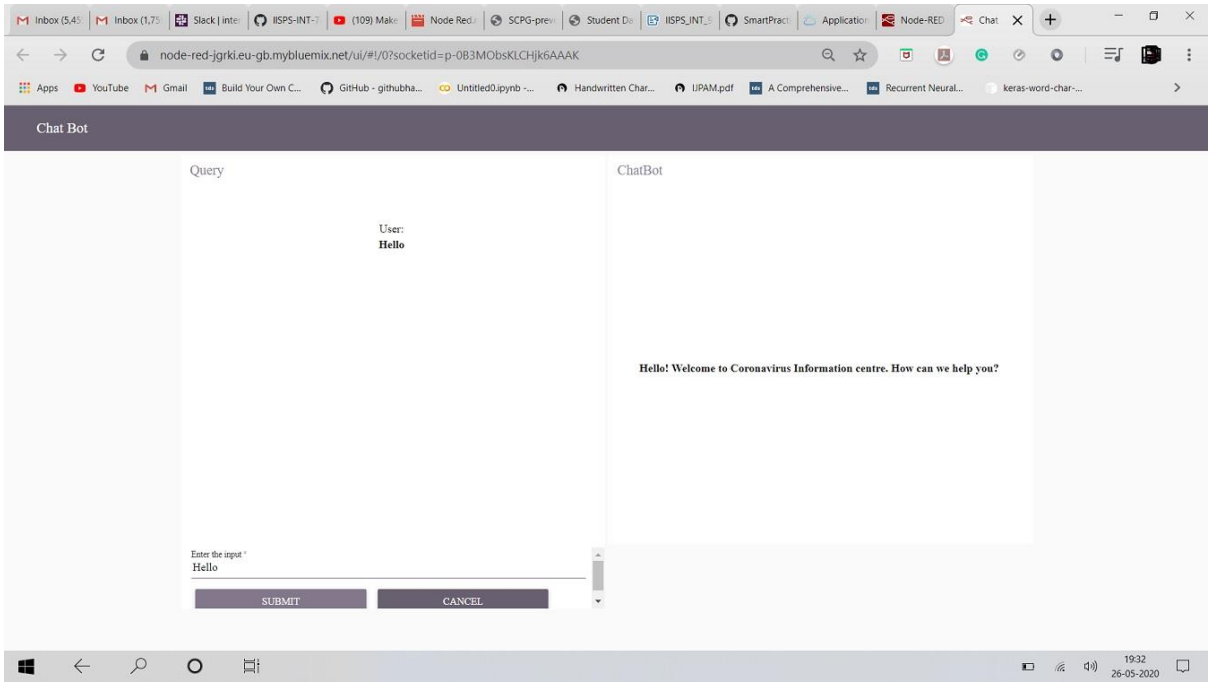


Figure 5.1. Output 1

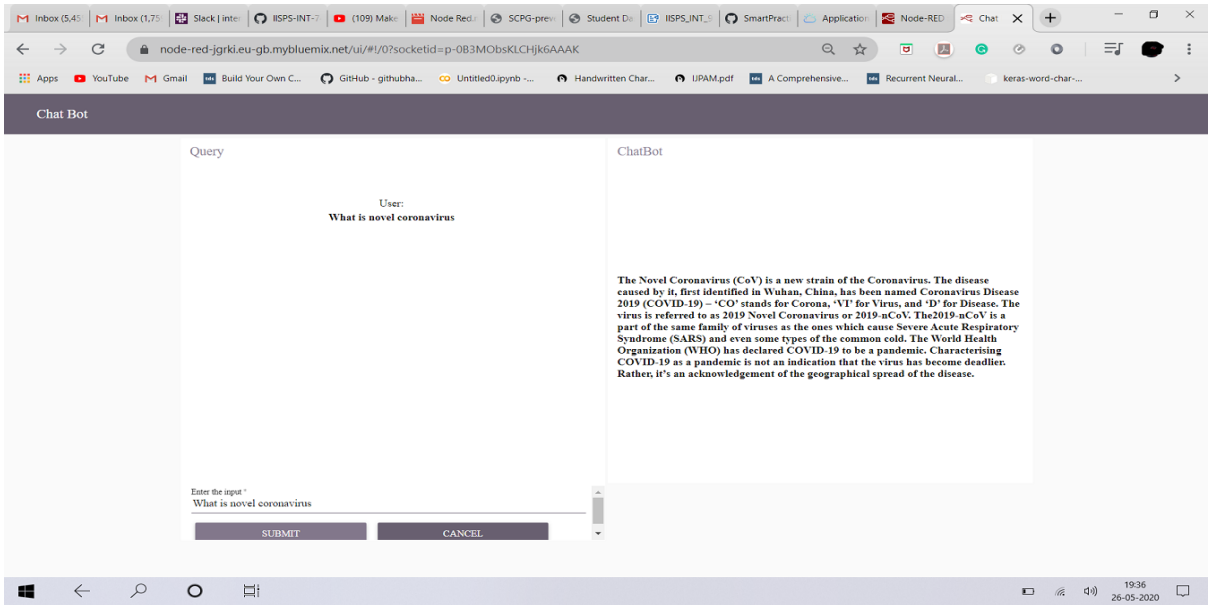


Figure 5.2. Output 2

6.ADVANTAGES & DISADVANTAGES

6.1.Advantages

- All required information about coronavirus in one application
- Reduces human intervention
- Time-efficient

6.2.Disadvantages

- Requires more live information
- Since this is a basic chatbot, information that is provided needs more sophistication
- Updation in current UI

7.CONCLUSION

This application is a basic chatbot for coronavirus information. It provides basic information like description, symptoms, prevention, and al basic facts.

8.FUTURE SCOPE

In the future, the bot can be improvised with live UI. Also, features like similar information, helpline contacts, Watson text to audio, extended user requests, a database to store and display recent chats, etc.

APPENDIX

Action 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'
      });
    }
    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);
  });
 });
}
```

REFERENCES

1. https://en.wikipedia.org/wiki/Chatbot
2. https://www.unicef.org/india/media/3231/file/COVID-Handbook-for-journalists.pdf
3. https://cloud.ibm.com/apidocs/discovery
4. https://cloud.ibm.com/apidocs/assistant/assistant-v2
5. https://smartinternz.com/

PROJECT CHECKLIST

Project Name: Intelligent Customer Help Desk

Kickoff Date: 12-05-2020

Client: SmartInternz

Project Manager: Sruthi Prabakaran P

Topic	Content
1. Contract	SmartInternz
2.Client Need	1. To develop intelligent customer help desk 2. To answer operational questions that usually are diverted to a real person 3. To use smart document understanding feature to improve answers
3.Project Plan	As specified in the dashboard
4.Team members roles	Sruthi - Project engineer
5.Scope of work	Used in regular business chatbots to answer customer queries
6.Project schedule	29 days starting from 12-05-20
7. Project Budget	nil
8.Project reviews	-
9.Project Expectations	a fully performing chatbot that can answer extended queries other than basic inquiry

PROJECT KICK-OFF AGENDA

Project Name: Intelligent Customer Help Desk

Kickoff Date: 12-05-2020

Topic	Content
1. Introduce and Welcome Team members	Sruthi - Project Engineer
2.Discuss the project background	1. Starting the project 2. Change the project path according to requirements 3. Achieving milestones without delays will help reach success
3. Identify stakeholders	1. Customers reaching out chatbots for inquiries 2. Project manager(myself) 3. Mentors
4.review project objectives	An Intelligent chatbot application that can answer technical questions as in user manual besides basic inquiries about the store like opening and closing hours
5.Review team member roles & responsibilities	Sruthi - Develop the first-level application
6.Review other potential issues, risks, questions, and concerns	Potential problems - an overload of the bot, ambiguous questions, unwanted questions Risks - Missing deadlines, Server crash, Hacks, Intrusion, System Hang
7. Identify next steps and timing	weekly meeting with mentors