1. **INTRODUCTION**

**1.1 Overview**

IBM cloud computing is a set of cloud computing services which can be used to perform different tasks and create a number of web applications. The project requirement was to create a **Smart Assistant for Public Parks using IBM Watson**. 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.

Here I have used the **IBM Watson Assistant**, **Speech to Text Service** and **Text to Speech Service** to create the back-end flow. For the front-end section, I have used the **Node-Red Service** to create a dashboard for the users.

**1.2 Purpose**

**Smart Assistant** solves the problems of the user by providing appropriate answers to their queries. It reduces the effort of the user. Here the smart assistant will give basic information about public parks to the users.

1. **LITERATURE SURVEY**

**2.1 Existing Problem**

Public parks are built by the government for the recreation of the citizens. Usually, people **do not have proper information** related to these parks. Even if they search on google for such information, several websites pop up with either **varying or incomplete information**. As a result, this confuses the user. So ultimately they get improper information.

Another such scenario is, if someone goes to a park, all the details are either given by a helper assigned by the authorities of the park or no details is given at all. In some of the parks, certain boards are installed where information is written regarding the attractions of the park. Now in the 21st century, people hardly stand and read anything. So this becomes problematic.

**2.2 Proposed Solution**

The world is growing digital day by day. Hence, the problem can be solved by using a chatbot that is a smart assistant which is an **Artificial Intelligence (AI)** conversational agent.

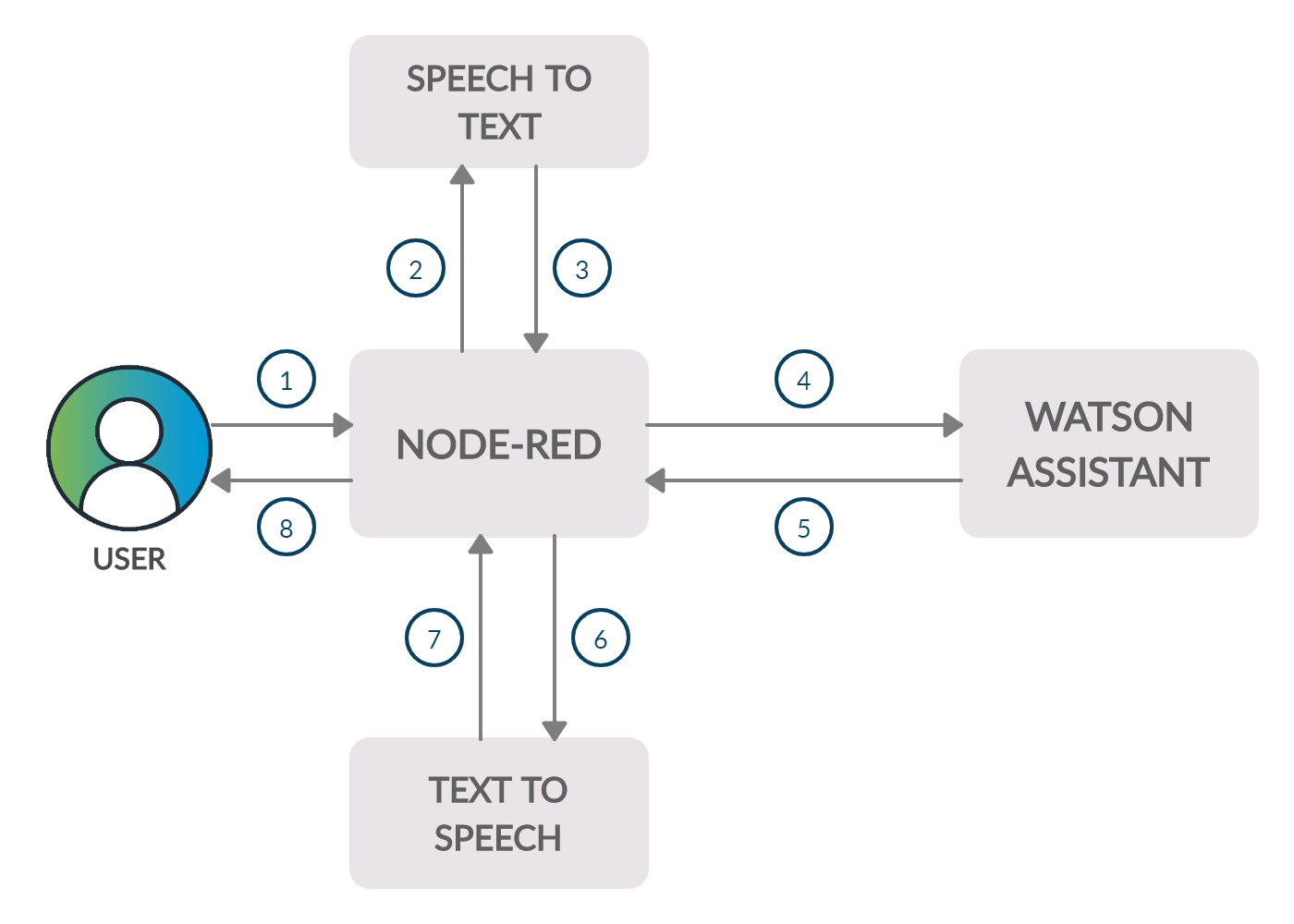
A chatbot communicates via text or voice commands. It communicates and performs basic tasks through messaging application, mobile applications or web applications. The **IBM Watson Assistant** allows us to train the chatbot according to our needs.

Here, **Park Assistant** is the chatbot which will solve all the problems related to public parks. It receives commands in the form of speech and will also answer in the same form. This chatbot will be effortless to use. The only task of the user will be to ask questions. The smart assistant will do the rest of the work.

1. **THEORETICAL ANALYSIS**

**3.1 Block Diagram**

The block diagram of the Park Assistant is shown below:



The flow of the block diagram is explained as follow :

* 1. The user visits a voice-enabled Node-RED website and asks a question.
  2. Node-Red records the speech and sends it to the Speech to Text Service of IBM Cloud.
  3. The Speech to Text Service converts the speech file into text and sends it back to Node-RED.
  4. Node-RED sends the text that is the question of the user to the Watson Assistant.
  5. Watson Assistant uses natural language understanding to extract the intents and entities of the user's question and then gives the answer to Node-RED in text format.
  6. Node-RED then sends the text to the Text to Speech Service of IBM Cloud.
  7. Text to Speech Service converts the answer given by the assistant in text format to a speech file and sends it to Node-RED.
  8. Node-Red plays the speech file and thus the user listens to the answer of the chatbot.

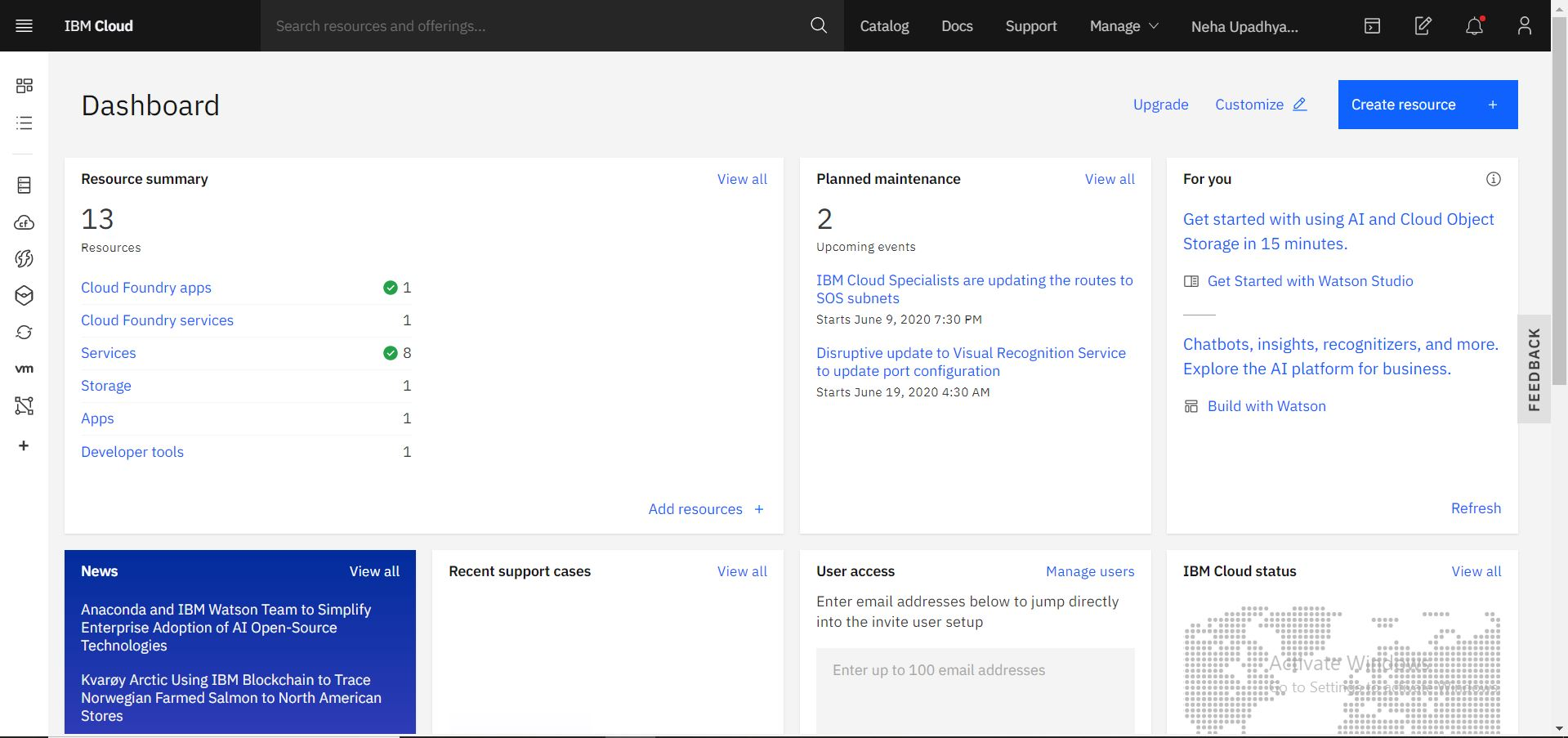
**3.2 Software designing**

The software used in this project are as follows :

**IBM Cloud -** IBM Cloud is one of the cloud computing platforms that offers both **PaaS (Platform as a Service)** and **IaaS (Infrastructure as a Service)**.

With IBM Cloud IaaS, organizations can deploy and access virtualized IT resources such as compute power, storage and networking over the internet. For compute, organizations can choose between bare-metal or virtual servers.

With IBM Cloud PaaS, which is based on the open source cloud platform, developers can use IBM services to create, manage, run and deploy various types of applications for the public cloud, as well as for local or on-premises environments. IBM Cloud supports various programming languages, such as **Java, Node.js, PHP and Python** and extends to support other languages.



**IBM CLOUD DASHBOARD**

**Python -** The back-end programming is done in Python. It is a **general purpose, interpreted, high level language**. It is dynamically typed, that is, it identifies the variable on the basis of what kind of data we have assigned to the variable. Python is a very user friendly language as it is easy to understand and the syntax is also quite simple. It is open source and readily available.

1. **EXPERIMENTAL INVESTIGATIONS**

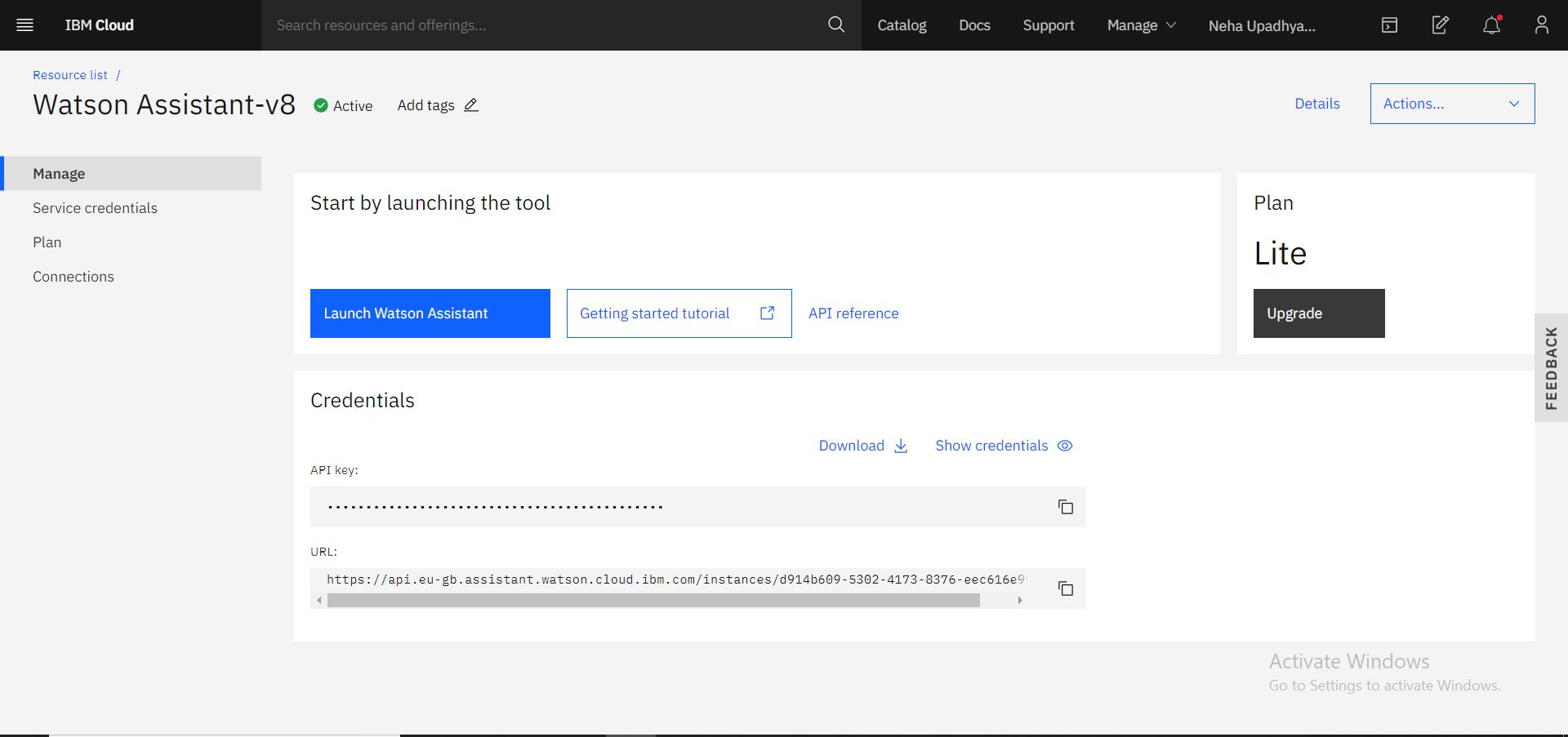
The **Park Assistant** is an integration of the **IBM Watson Assistant**, **Speech to Text Service**, **Text to Speech Service** and **Node-RED Service**.

**IBM Watson Assistant** - This allows us to build our own live chatbot which can be deployed on any device, application or channel. The user and the chatbot interact through a virtual assistant or a custom application. The user gives input which is then routed to the dialog skill by the assistant. The dialog skill interprets the user input further and then generates a conversation flow accordingly.Intents and entities are identified by the assistant.

**Intents** helps the assistant to understand the intention of the user. By recognizing the intent expressed in a customer's input, the **Watson Assistant** service can choose the correct dialog flow for responding to it. IBM Watson Assistant provides some prebuit intents which can be used for building an assistant. For instance, #General\_Greetings, this intent includes all the different types of greeting ways a user can input while using the assistant. Other than the prebuilt intents, we can also create out own intents by giving a specific name to it and then including the examples of different intentions of the user.

**Entities** allows the assistant to search the values as well as the synonyms of those values to understand in a better way. Recognizing entities in the user's input helps us to craft more useful, targeted responses. There are different types of entities like - **synonym entity, pattern entity, contextual entity** and **system entity**. All these types of entities can be used to train our chatbot in an efficient manner.

After adding the intents and entities to our assistant, we can build its dialog skill. The **dialog** recognizes the intents and gives the response accordingly. The dialog flow starts from the top node of the tree to the last node. However if any condition is recognized then it triggers the node as mentioned in the condition and continues with the flow.

 **IBM WATSON ASSISTANT SERVICE**

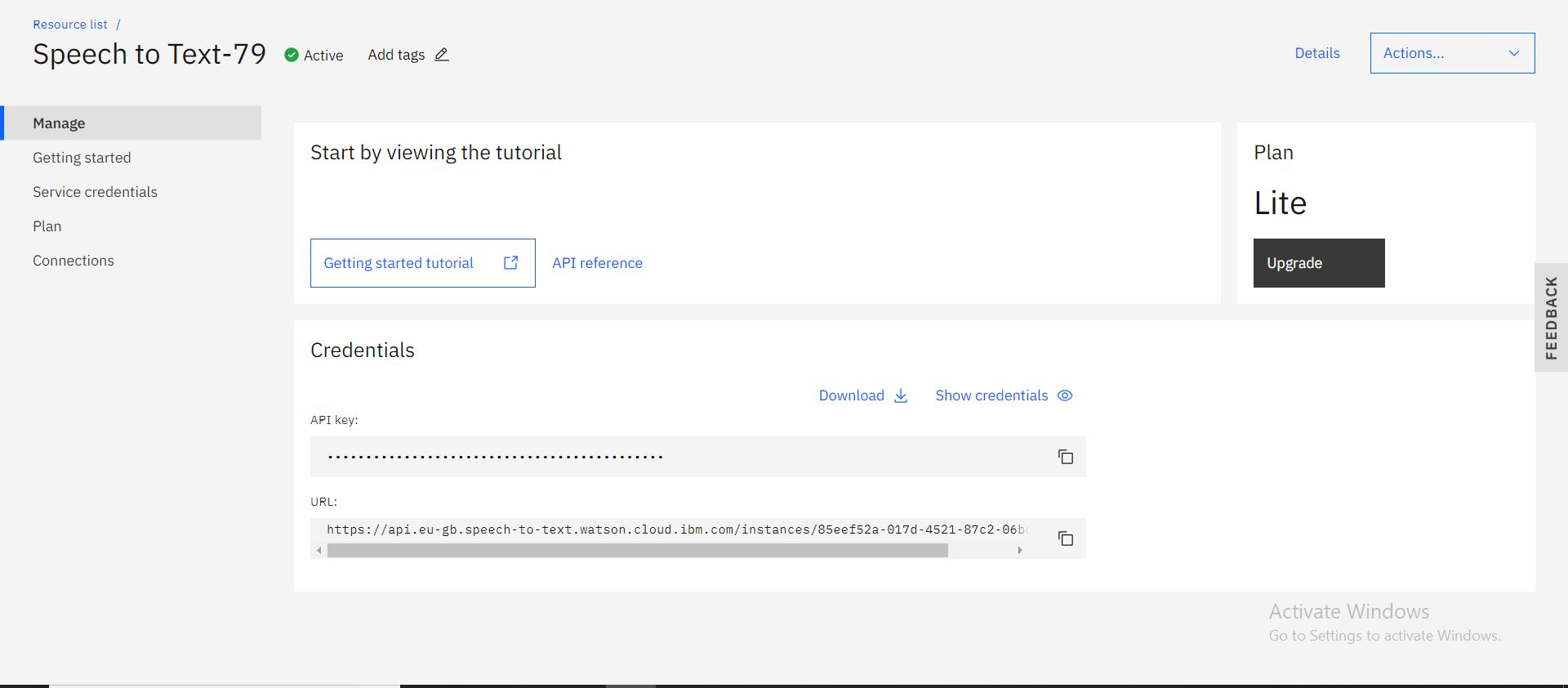
**Speech to Text Service** - This allows us to change the speech files to text. It provides the speech transcription capability for our applications. It continuously updates and refines its transcription as it receives more speech.

The input features include the following:

* It can take voice input in Ogg or Web Media (**WebM**) audio with the Opus or Vorbis codec, **MP3 (or MPEG)**, Waveform Audio File Format (**WAV**), Free Lossless Audio Codec (**FLAC**), Linear 16-bit Pulse-Code Modulation (**PCM**), **G.729**, **A-Law**, mu-law (or **u-law**), and basic audio.
* It can take the speech **input in a number of languages** like US English, UK English, German, French, Spanish and many more.
* The **speech activity detection** can suppress the background noises and can only transcribe the relevant part of the voice input.

The output features includes the following:

* `It gives the output in a **number of languages**.
* It **identifies spoken phrases** that match specified keyword strings with a user-defined **level of confidence**.
* It return **timestamps** for the start and end of each word of a transcript.
* It can **smartly format** the input accordingly and gives output with a certain level of confidence.

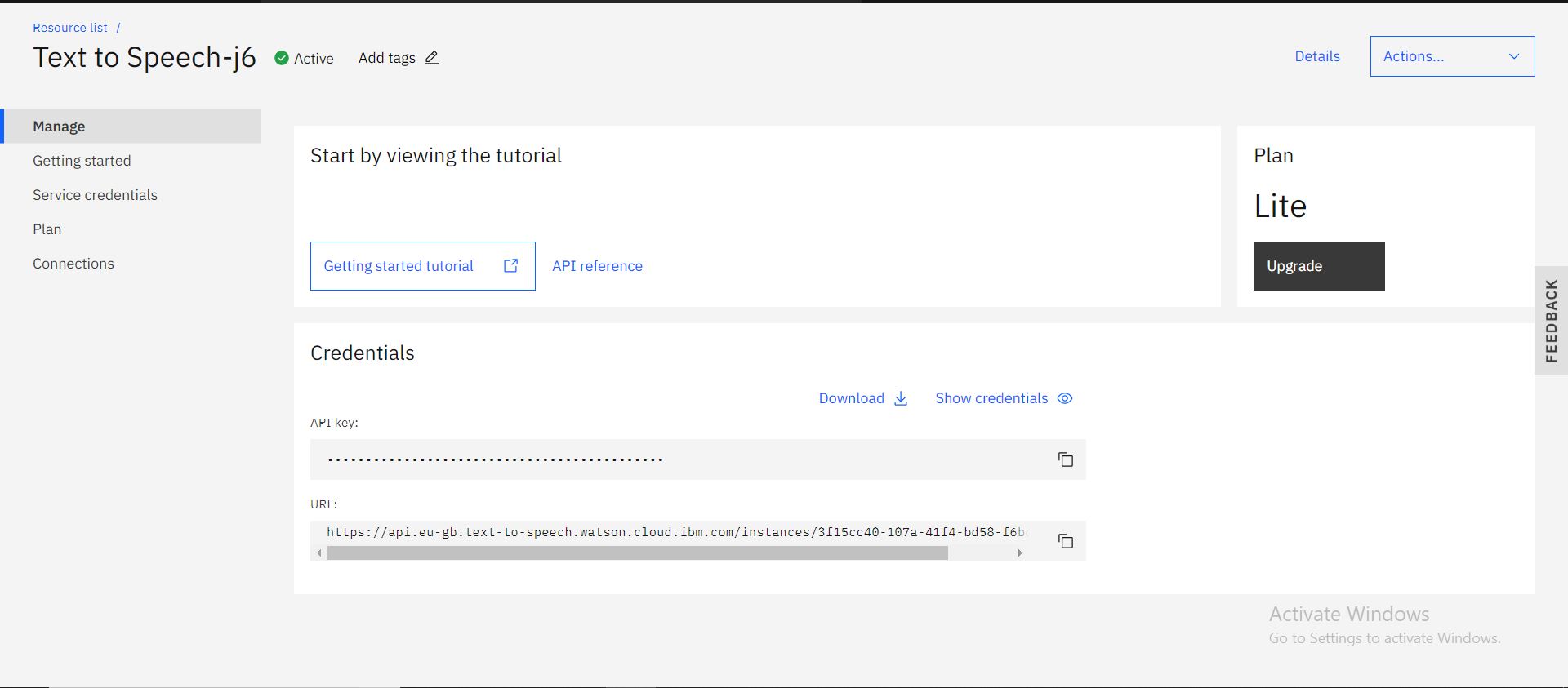


**SPEECH TO TEXT SERVICE**

**Text to Speech Service -** This allows the conversion of text input to speech files as output. It provides an **application programming interface (API)** that uses IBM's speech-synthesis capabilities to convert written text to natural-sounding speech.

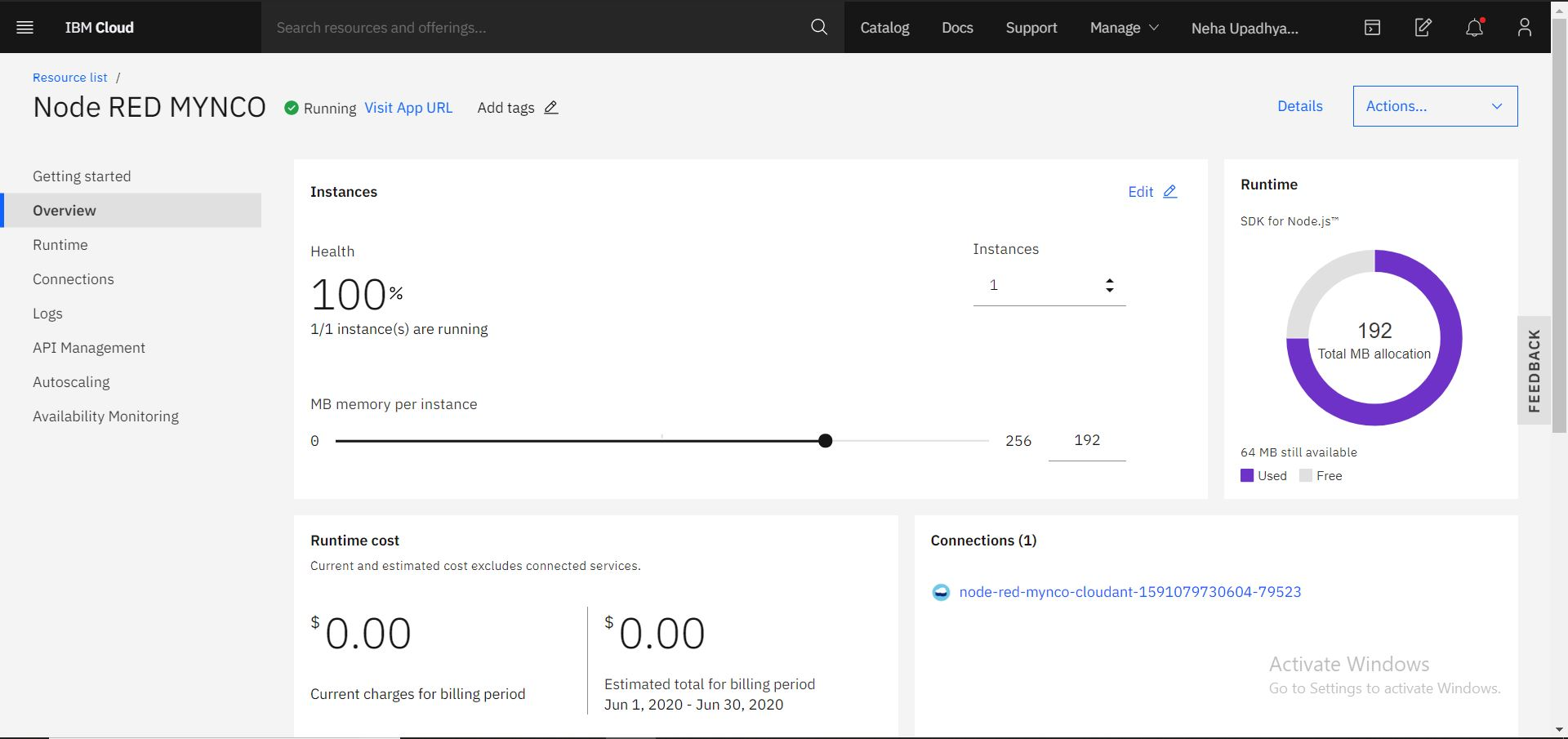
The features of Text to Speech Service are as follows:

* + - It produces audio in Ogg or **WebM** with the Opus or Vorbis codec, **WAV**, **FLAC**, **MP3 (MPEG)**, **l16 (PCM)**, **mulaw**, or basic format.
    - It takes input in **Speech Synthesis Markup Language (SSML)**.
    - The output can be provided in **various languages** like English, Spanish, German, Japanese, etc.
    - It also includes **Expressive SSML** which means that it can change its speaking style according to our needs.
    - This service also produces the output including **voice transformation**.

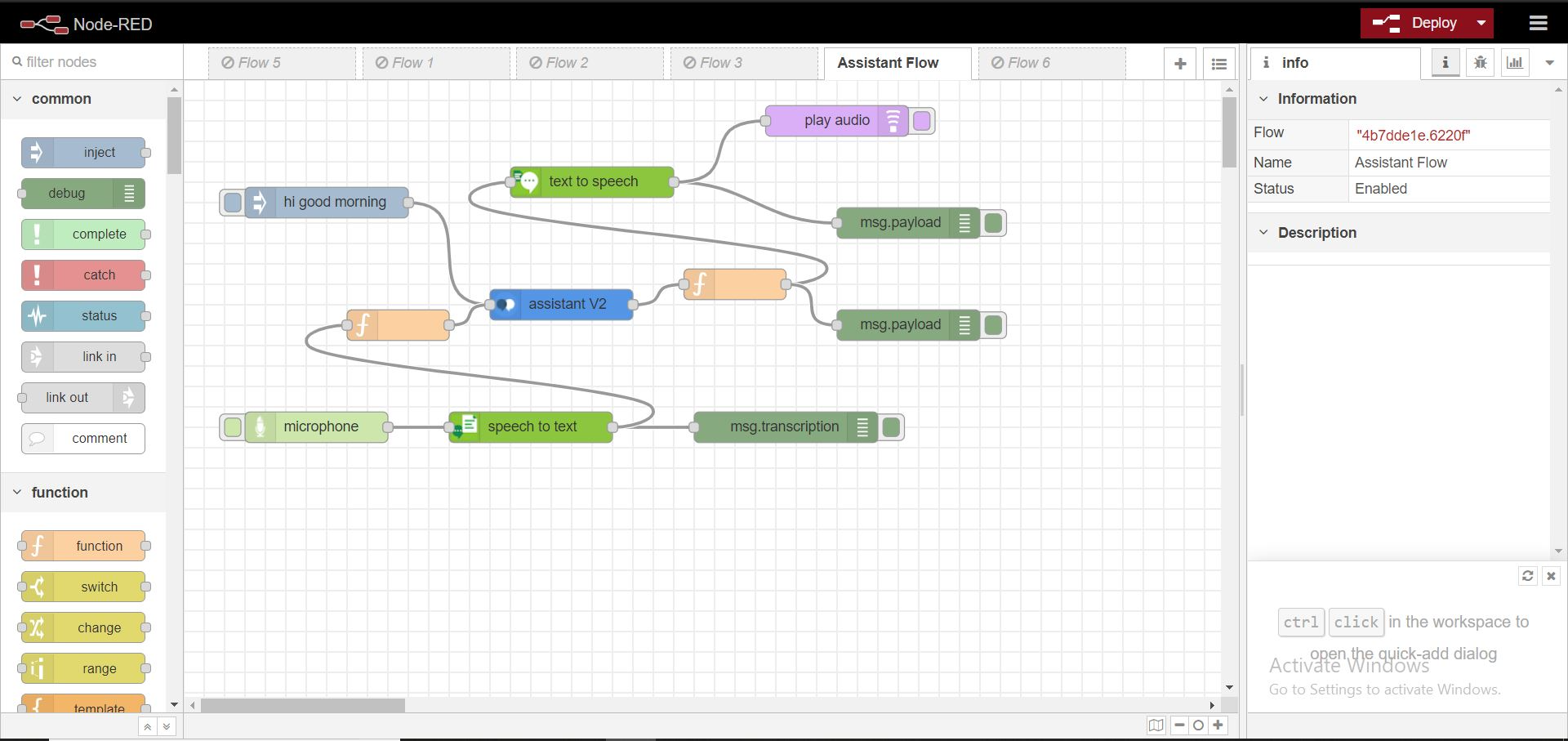
 **TEXT TO SPEECH SERVICE**

**Node-RED Service -** The Node-RED is a **flow-based programming tool** which helps to conect various elements together using nodes including hardware devices, online services and APIs. It is built on **Node.js**. We can install nodes from the manage palette and then use these nodes by joining them with wires. Finally after deploying the whole network can run.

The **Node-RED** networks can be shared as **JSON** files. It also includes Watson IoT nodes so that we can connect with the Watson Assistant using this service. Its lightweight nature makes it ideal to run at the edge of the network as well, such as on the Raspberry Pi, and other hack-friendly platforms.



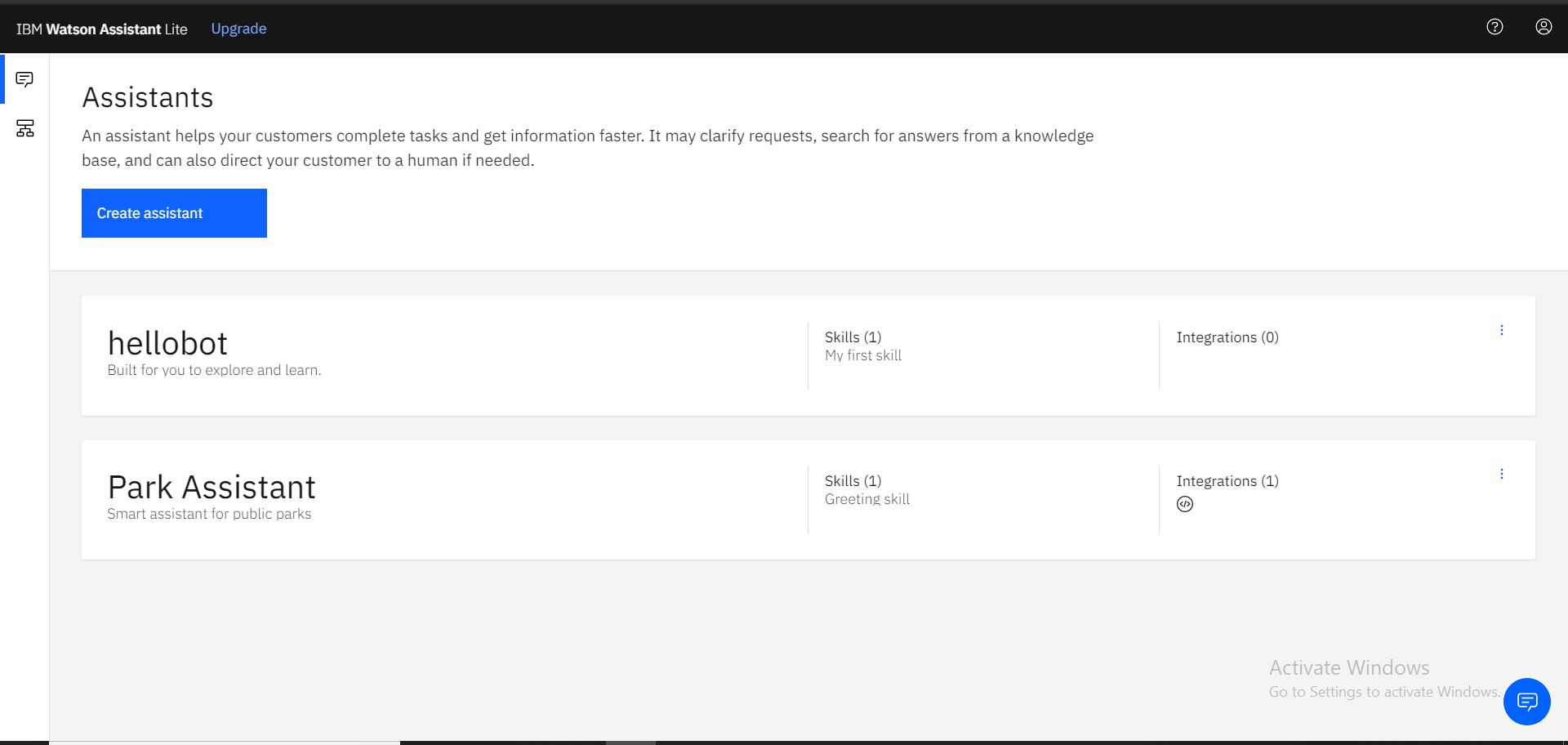
**Node-RED**



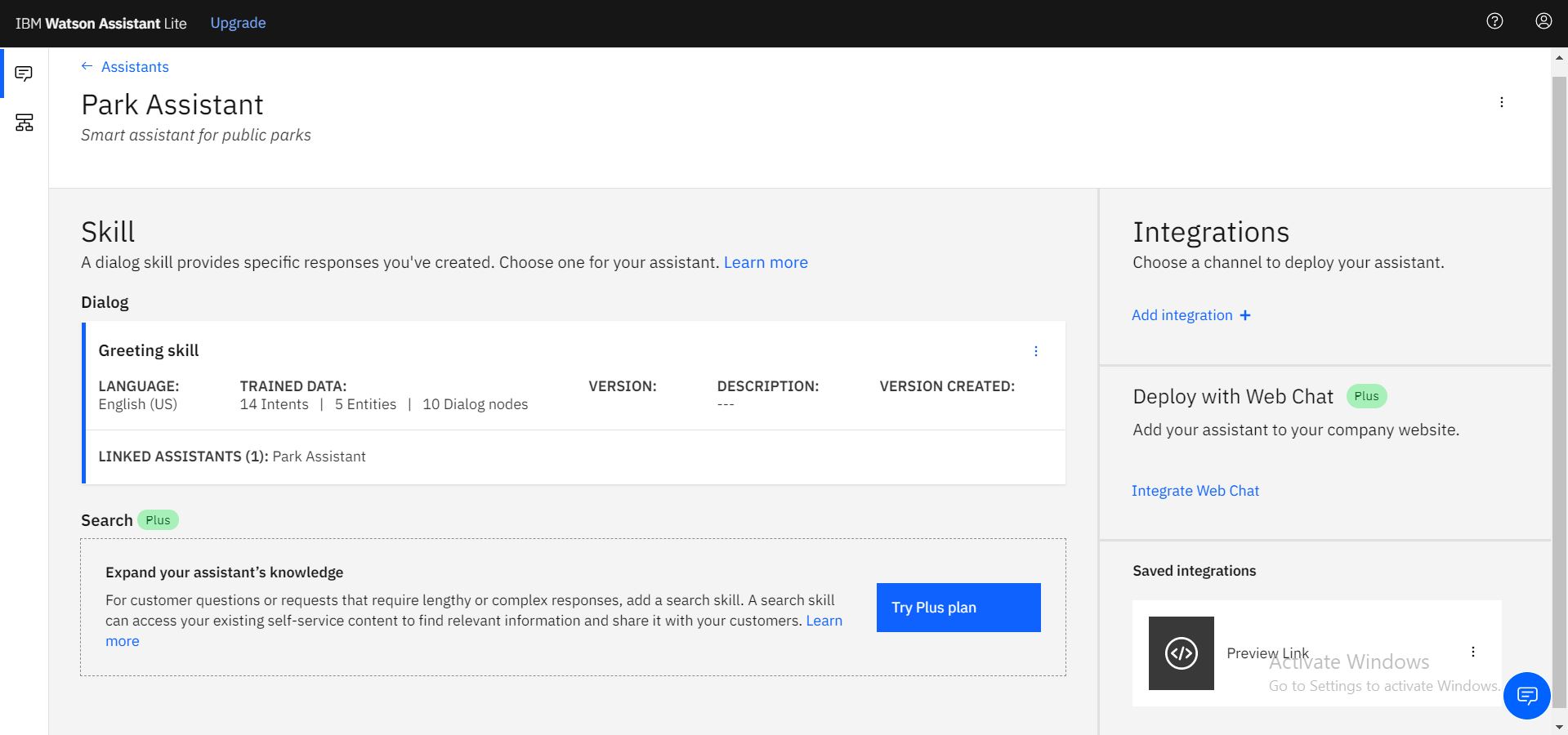
**Node-RED WORKSPACE**

1. **RESULT**

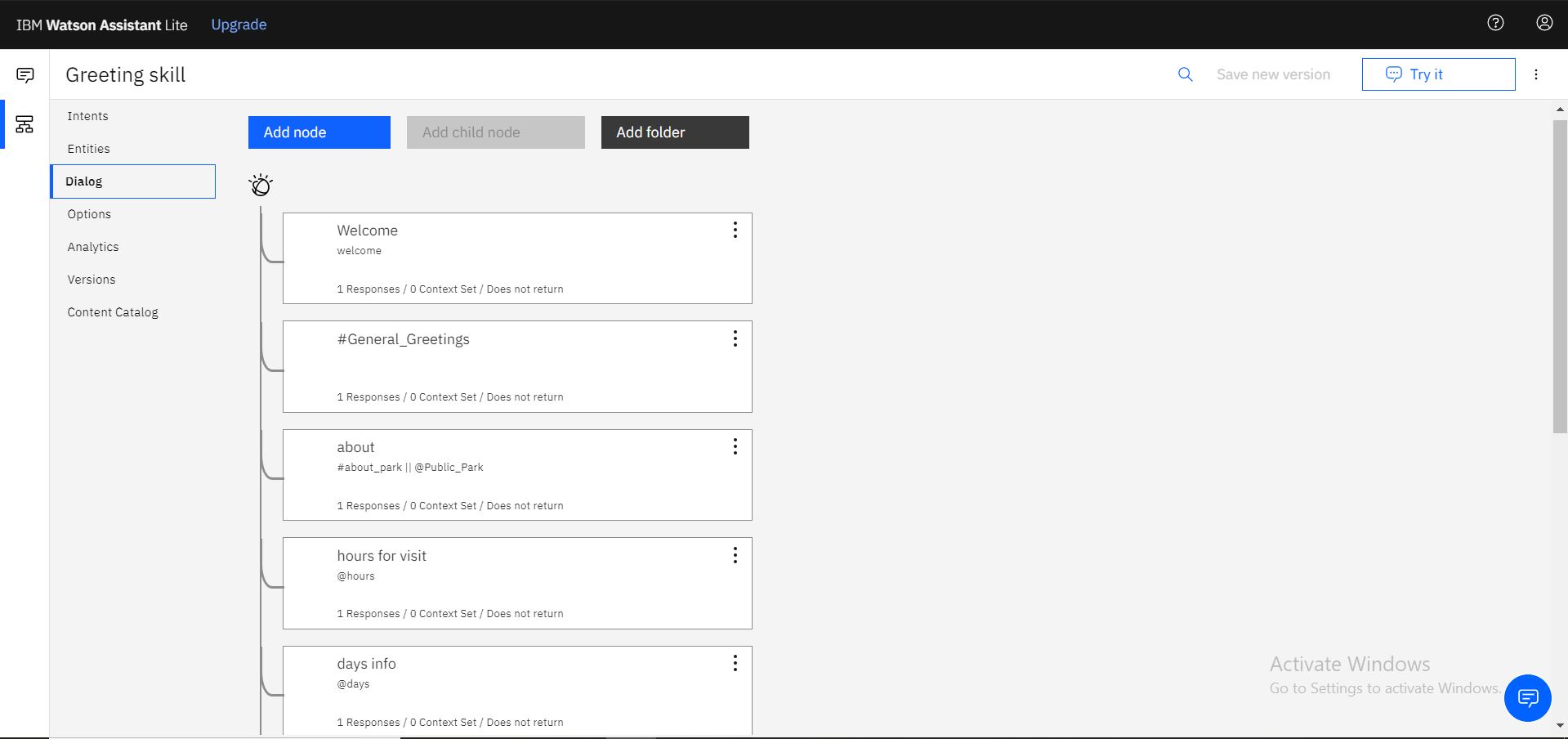
After writing the python codes, the flow needs to be created in the Node-RED platform where all the services are integrated togrther. In the Watson Assistant service, the intents and entities are created for the chatbot and then the bot is trained to answer in the specific manner. That is shown in the images below:



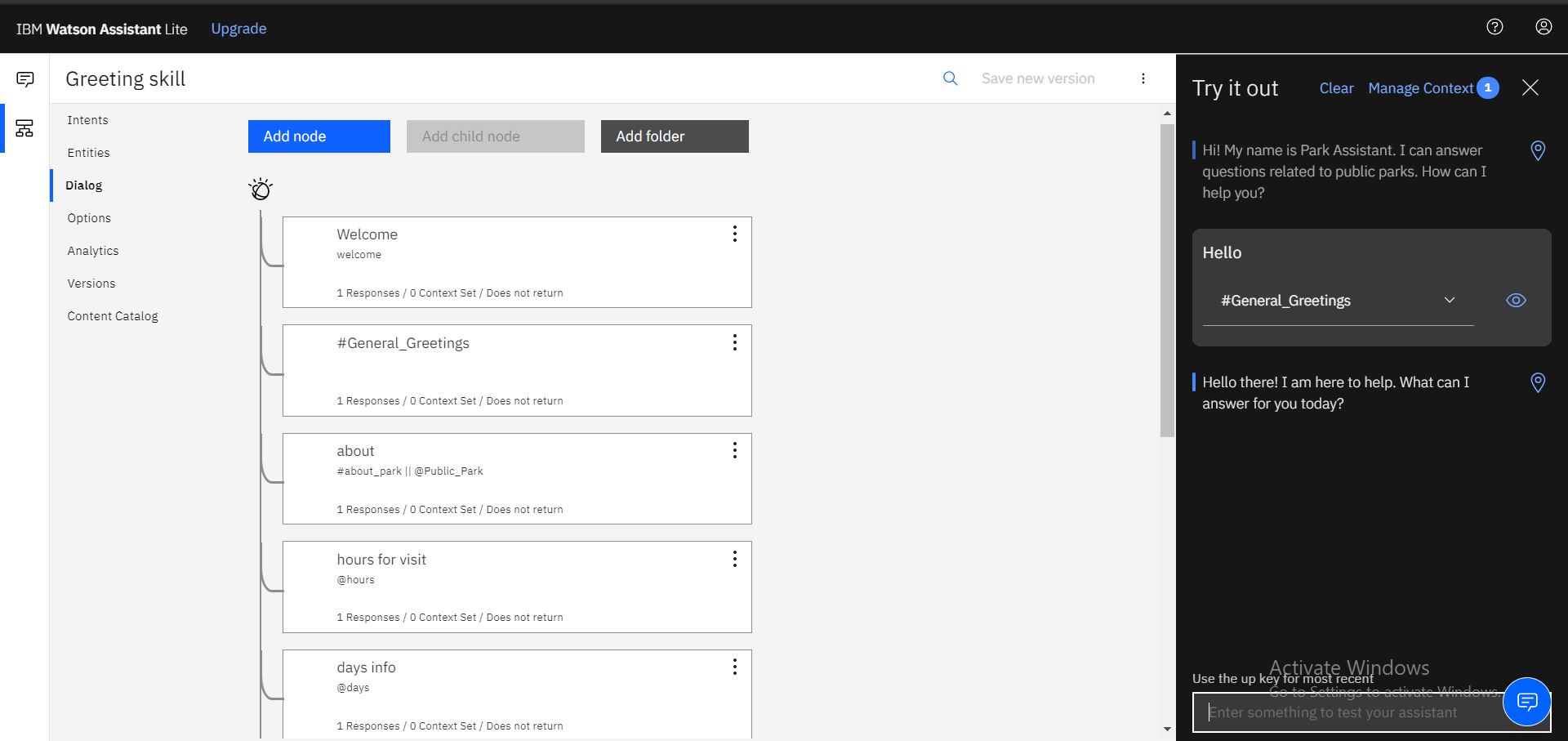
**CREATION OF PARK ASSISTANT**



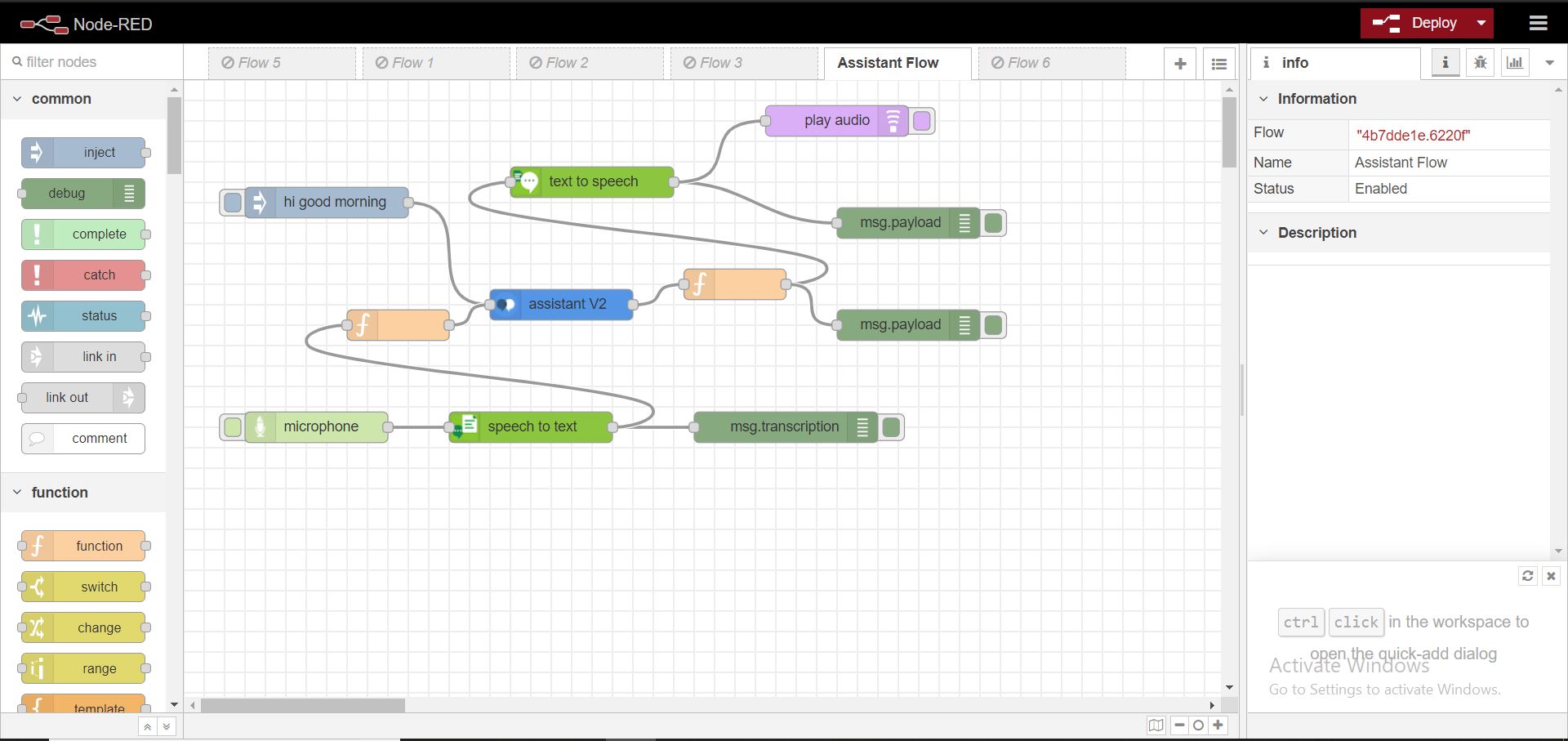
**SKILLS OF PARK ASSISTANT**



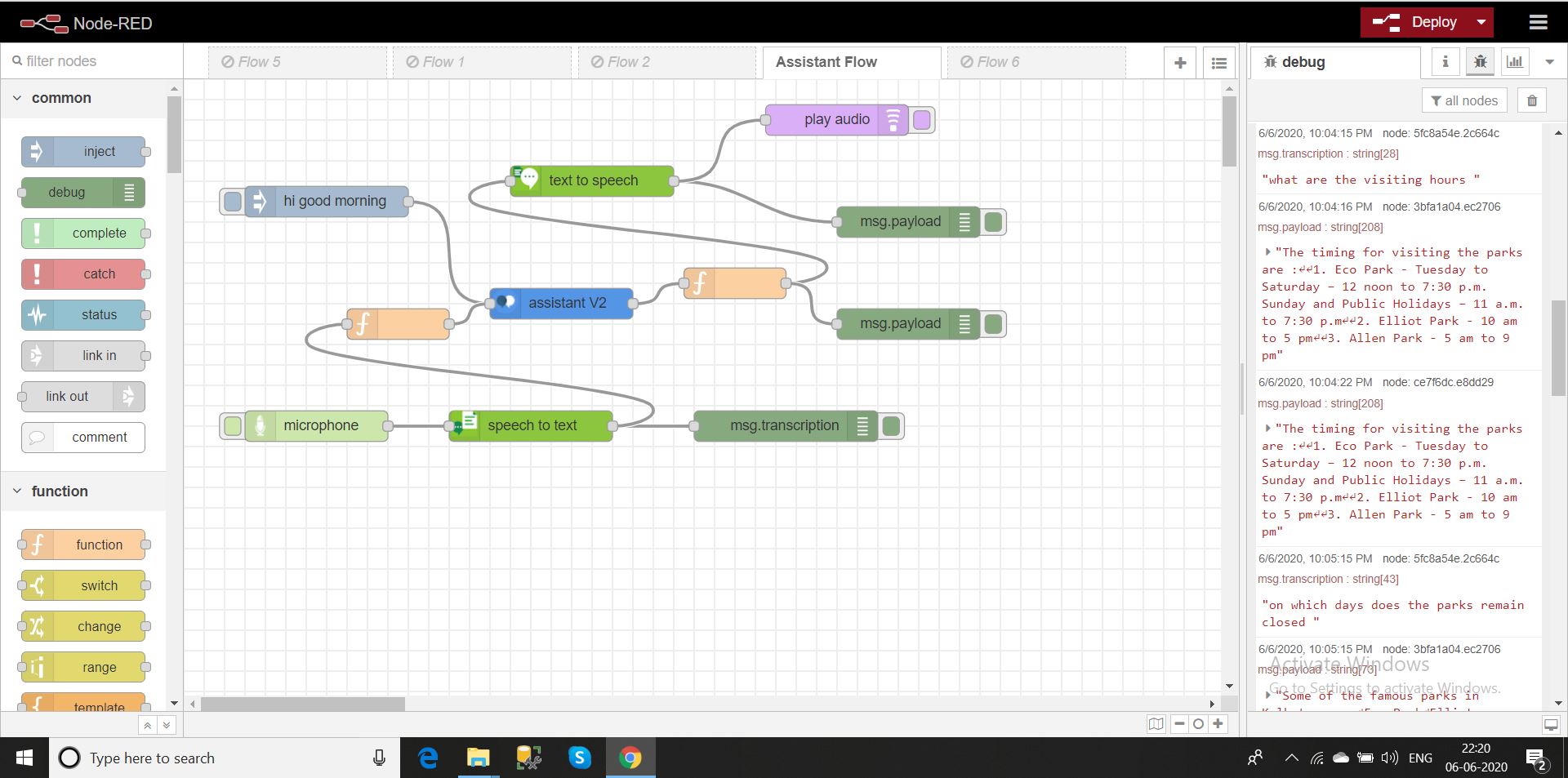
**DIALOG FLOW OF THE ASSISTANT**

 **OUTPUT SHOWING HOW THE PARK ASSISTANT RESPONDS**

After training the chatbot, Node-RED flow is designed. The image of the flow is shown below:



**Node-RED FLOW DIAGRAM**



**OUTPUT OF THE Node-RED FLOW**

After joining all the nodes, the flow is deployed. The input needs to be provided by the microphone node in the form of speech and the output is recieved in the form of speech as well. The transcribed question of the user and the response of the assistant are all displayed in the console of the Node-RED.

1. **ADVANTAGES & DISADVANTAGES**

The **advantages** of using the **Park Assistant** are as follows:

* + - Proper information is provided to the user.
    - User does not need to give any type of effort.
    - Readily available.
    - Easy to use as the user has to talk to the assistant.
    - Pocket friendly

On the other hand, the **disadvantages** are as follows:

* + - Minimum internet connection is required.
    - The input should be given properly so that the assistant can understand the question being asked.
    - The assistant can answer only those questions about which he has been trained.

1. **APPLICATIONS**

The **applications** of **Park Assistan**t are as follows:

* + - It can be used by the owners of a new park and help this to **advertise** their park. This can help them to **gather more customers**.
    - It can be used by **blind people**. They can visit such parks only if they know whether it is friendly for them or not and Park Assistant will be sure to give that information in the **form of speech**.
    - This assistant will be very **helpful for the tourists** as they will not be misguided whenever they travel to a new place. If they wish to visit any public parks in that area, they would get proper information from Park Assistant.

1. **CONCLUSION**

In the conclusion, I would like to say that **IBM Cloud** provides a very efficient platform for creating our own chatbots by following some very simple steps. The proposed solution for the given problem is to introduce a smart way to get information regarding the public parks. So a **smart assistant** is the only answer for this problem. It is easy to use and requires no extra input from the user. Thus it can be used by any age group as well. The versatile programming language, **Python**, makes it easier for the budding developers to create something that can be used in our real lives.

**10. FUTURE SCOPE**

The **future scope** of the **Park Assistant** are given below:

* + **More information** realted different parks can be added.
  + The chatbot can be trained to answer a **wide range of questions**.
  + The chatbot can also include **location** details of the parks as well as the user.
  + Sites for **booking the tickets** can be shared using the APIs.
  + Information related to **nearby attractions** can also be added for the users.

**11. BIBLIOGRAPHY**

During this project I have referred to certain websites. The links of those websites are given below:

* + - <https://cloud.ibm.com/docs>
    - <https://nodered.org/>
    - <https://developer.ibm.com/components/node-red/tutorials/create-a-voice-enabled-covid-19-chatbot-using-node-red/>
    - <https://developer.ibm.com/recipes/tutorials/getting-started-with-watson-iot-platform-using-node-red/>

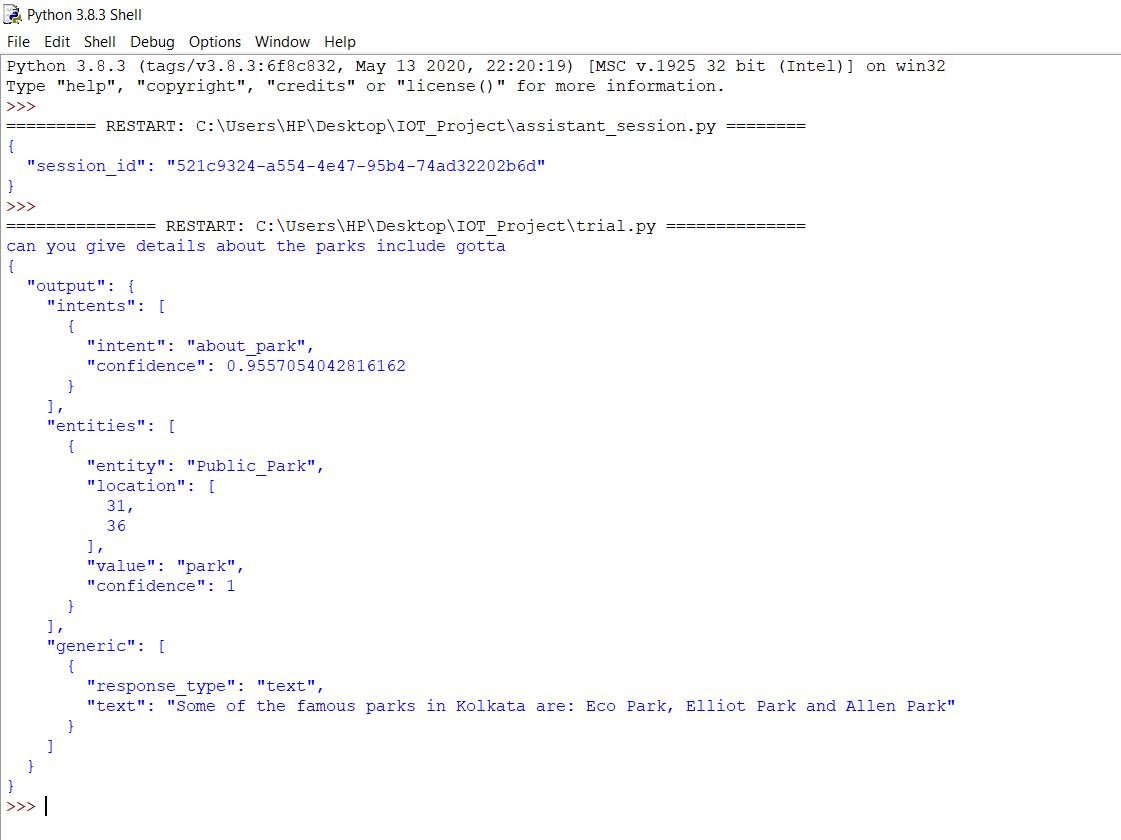
**12. APPENDIX**

**A. SOURCE CODE**

The source code used for this project are attached below:

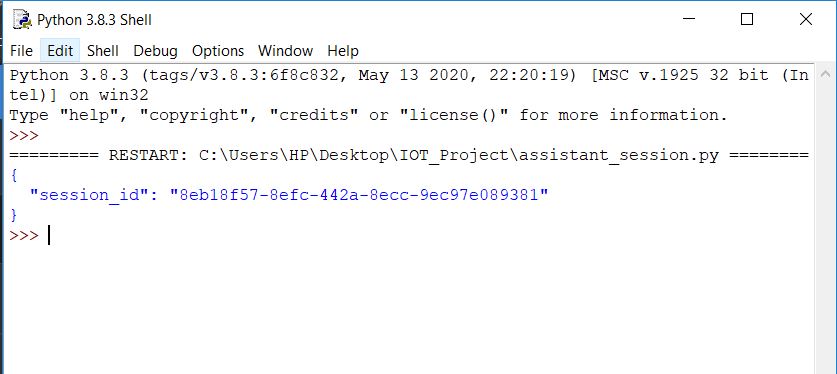
1. import json
2. from os.path import join, dirname
3. from ibm\_watson import AssistantV2, TextToSpeechV1, SpeechToTextV1
4. from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator
5. from playsound import playsound
6. authenticator = IAMAuthenticator('jPh4eRtgQSzB3zwLey9crrmPALlg\_5si9sIYqEPGzpzG')
7. speech\_to\_text = SpeechToTextV1(
8. authenticator=authenticator
9. )
10. speech\_to\_text.set\_service\_url('https://api.eu-gb.speech-to-text.watson.cloud.ibm.com/instances/85eef52a-017d-4521-87c2-06bd0ebf82ef')
11. with open(join(dirname(\_\_file\_\_), './.', 'about\_parks.mp3'),
12. 'rb') as audio\_file:
13. speech\_recognition\_results = speech\_to\_text.recognize(
14. audio=audio\_file,
15. content\_type='audio/mp3',
16. ).get\_result()
17. #print(json.dumps(speech\_recognition\_results, indent=2))
18. str = ""
19. while bool(speech\_recognition\_results.get('results')):
20. str=speech\_recognition\_results.get('results').pop().get('alternatives').pop().get('transcript')+str[:]
21. print(str)
22. authenticator = IAMAuthenticator('a4AqChKZqBJAKxuSH28TtAYQe0iuy2ve1gikX0ZQjdJX')
23. assistant = AssistantV2(
24. version='2020-04-01',
25. authenticator = authenticator
26. )
27. assistant.set\_service\_url('https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/d914b609-5302-4173-8376-eec616e99f8f')
28. response = assistant.message(
29. assistant\_id='f464c096-684c-4a7e-9de2-d8d8a5748a16',
30. session\_id='4d3a3d8d-7349-4b5b-96c0-fb03a6034140',
31. input={
32. 'message\_type': 'text',
33. 'text': str
34. }
35. ).get\_result()
36. print(json.dumps(response, indent=2))
37. authenticator = IAMAuthenticator('MMVDKZdT9xYQKbwFeSHmEWYLcDwBZUFturfiB6IcV4z0')
38. text\_to\_speech = TextToSpeechV1(
39. authenticator=authenticator
40. )
41. text\_to\_speech.set\_service\_url('https://api.eu-gb.text-to-speech.watson.cloud.ibm.com/instances/3f15cc40-107a-41f4-bd58-f6bd7f3a88e4')
42. with open('bot.mp3', 'wb') as audio\_file:
43. audio\_file.write(
44. text\_to\_speech.synthesize(
45. json.dumps(response["output"]["generic"][0]["text"]),
46. voice='en-US\_AllisonV3Voice',
47. accept='audio/mp3'
48. ).get\_result().content)
49. playsound('bot.mp3')

The output is shown below:

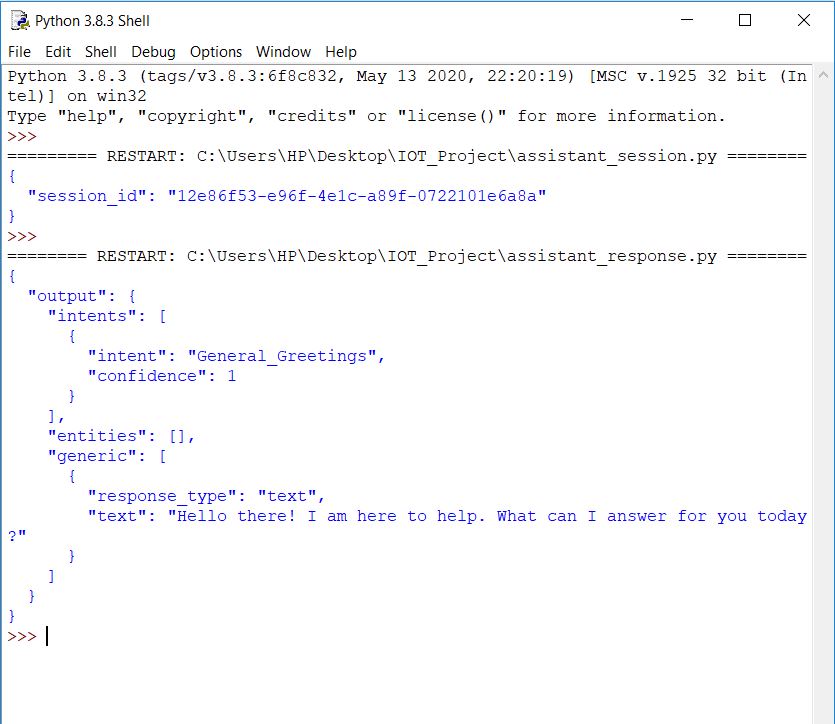


The code snippet for Watson Assistant session and response are as follows:

1. import json
2. from ibm\_watson import AssistantV2
3. from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator
4. authenticator = IAMAuthenticator('a4AqChKZqBJAKxuSH28TtAYQe0iuy2ve1gikX0ZQjdJX')
5. assistant = AssistantV2(
6. version='2020-04-01',
7. authenticator = authenticator
8. )
9. assistant.set\_service\_url('https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/d914b609-5302-4173-8376-eec616e99f8f')
10. response = assistant.create\_session(
11. assistant\_id='f464c096-684c-4a7e-9de2-d8d8a5748a16'
12. ).get\_result()
13. print(json.dumps(response, indent=2))
14. import json
15. from ibm\_watson import AssistantV2
16. from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator
17. authenticator = IAMAuthenticator('a4AqChKZqBJAKxuSH28TtAYQe0iuy2ve1gikX0ZQjdJX')
18. assistant = AssistantV2(
19. version='2020-04-01',
20. authenticator = authenticator
21. )
22. assistant.set\_service\_url('https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/d914b609-5302-4173-8376-eec616e99f8f')
23. response = assistant.message(
24. assistant\_id='f464c096-684c-4a7e-9de2-d8d8a5748a16',
25. session\_id='6b662d98-7ab8-4e94-b866-219f3da15fac',
26. input={
27. 'message\_type': 'text',
28. 'text': 'Hello'
29. }
30. ).get\_result()
31. print(json.dumps(response, indent=2))



**OUTPUT OF ASSISTANT SESSION**



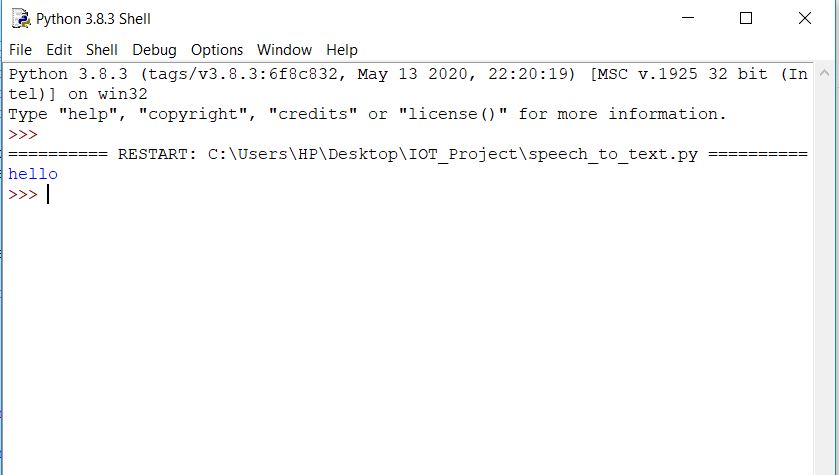
**OUTPUT OF ASSISTANT RESPONSE**

The code snippet for Text to Speech conversion is given below:

1. from ibm\_watson import TextToSpeechV1
2. from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator
3. from playsound import playsound
4. authenticator = IAMAuthenticator('MMVDKZdT9xYQKbwFeSHmEWYLcDwBZUFturfiB6IcV4z0')
5. text\_to\_speech = TextToSpeechV1(
6. authenticator=authenticator
7. )
8. text\_to\_speech.set\_service\_url('https://api.eu-gb.text-to-speech.watson.cloud.ibm.com/instances/3f15cc40-107a-41f4-bd58-f6bd7f3a88e4')
9. with open('bot.mp3', 'wb') as audio\_file:
10. audio\_file.write(
11. text\_to\_speech.synthesize(
12. 'Hello',
13. voice='en-US\_AllisonV3Voice',
14. accept='audio/mp3'
15. ).get\_result().content)
16. playsound('bot.mp3')

The code snippet for Speech to Text conversion is given below:

1. import json
2. from os.path import join, dirname
3. from ibm\_watson import SpeechToTextV1
4. from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator
5. authenticator = IAMAuthenticator('jPh4eRtgQSzB3zwLey9crrmPALlg\_5si9sIYqEPGzpzG')
6. speech\_to\_text = SpeechToTextV1(
7. authenticator=authenticator
8. )
9. speech\_to\_text.set\_service\_url('https://api.eu-gb.speech-to-text.watson.cloud.ibm.com/instances/85eef52a-017d-4521-87c2-06bd0ebf82ef')
10. with open(join(dirname(\_\_file\_\_), './.', 'hello.mp3'),
11. 'rb') as audio\_file:
12. speech\_recognition\_results = speech\_to\_text.recognize(
13. audio=audio\_file,
14. content\_type='audio/mp3',
15. ).get\_result()
16. #print(json.dumps(speech\_recognition\_results, indent=2))
17. str = ""
18. while bool(speech\_recognition\_results.get('results')):
19. str=speech\_recognition\_results.get('results').pop().get('alternatives').pop().get('transcript')+str[:]
20. print(str)

**OUTPUT OF SPEECH TO TEXT**