INDUSTRIAL TRAING PROJECT

PROJECT REPORT
ON
"Chatbot"

Govt. Polytechnic Manesar

Submitted to: Reena Mam

Submitted by: Neetu Negi

Branch: Computer Egg. (5^{th} sem)

Roll. No: 17007080029

PROJECT DESCRIPTION

A **chatbot** is a computer program that can converse with humans using artificial intelligence in messaging platforms. The most common predicted uses for chatbot's that consumers reported included getting quick answers to questions in an emergency (37%), resolving a complaint or problem (35%), and getting detailed answers or explanations (35%).

Goal:-

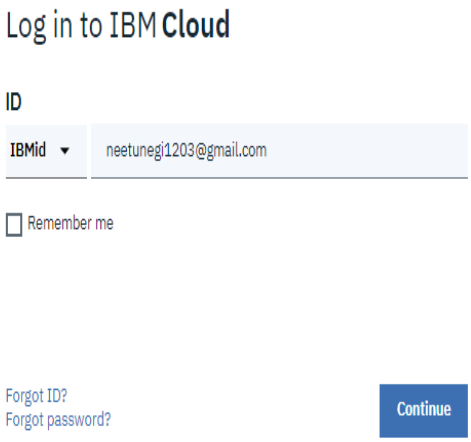
To make a chatbot using IBM Watson services. We are going to make a chatbot with text I/O, voice I/O, and GUI with voice input and voice plus text output: -

Prerequisite:-

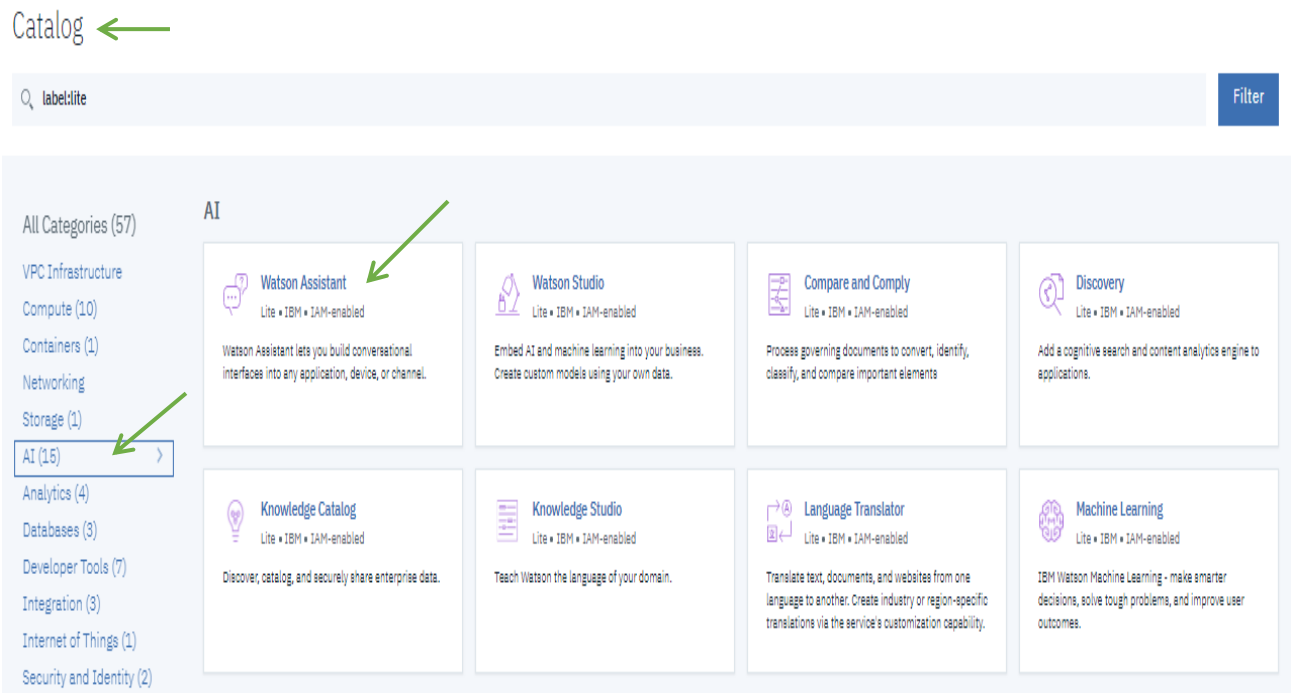
- IBM cloud account
- Libraries
    1. json
    2. ibm_watson

Steps:-

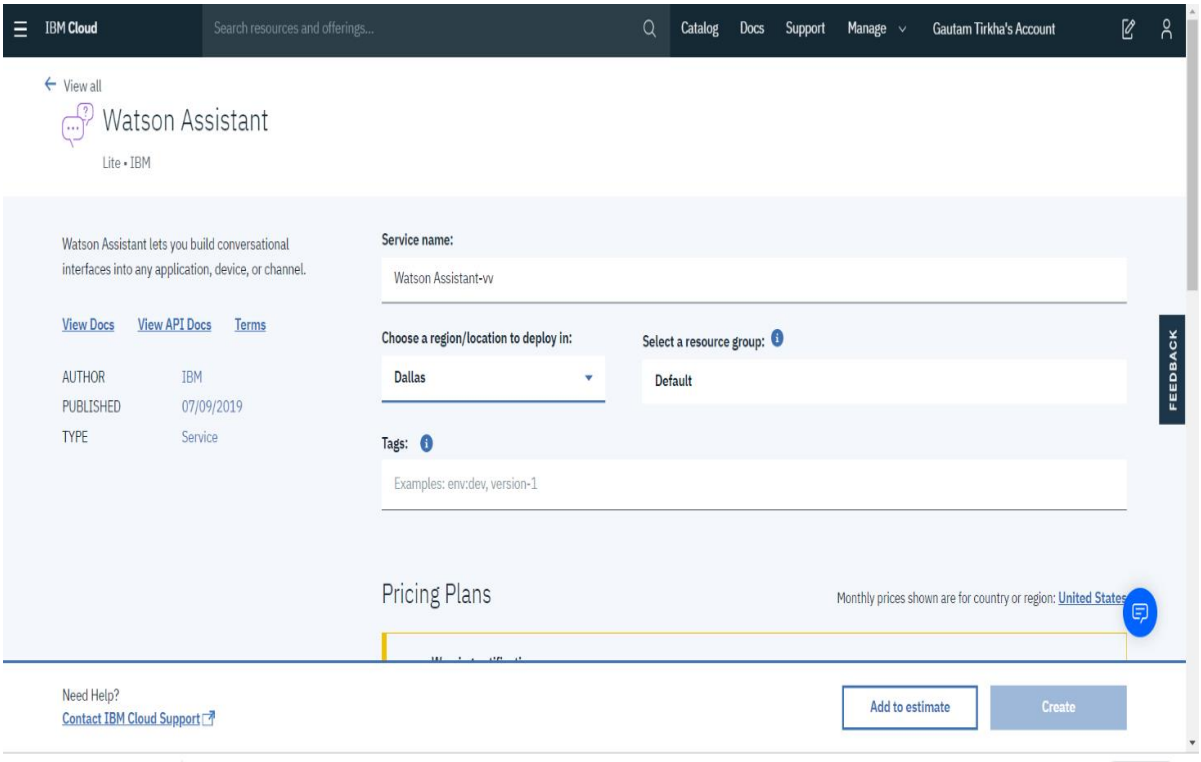
1. create an IBM Watson service on IBM cloud:-
    I. Go to https://cloud.ibm.com/ and sign in/sign up to create an IBM Watson service



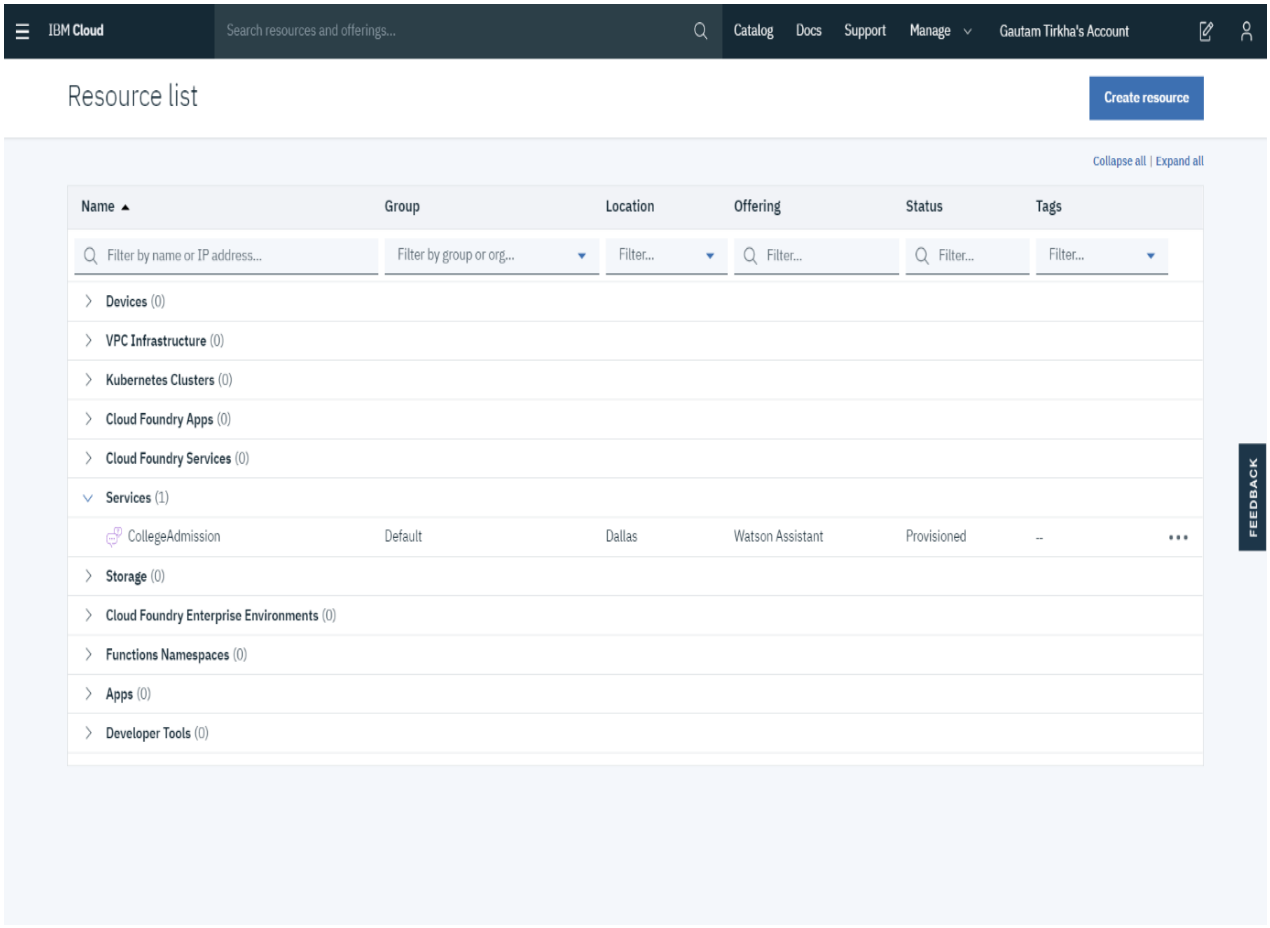
II. Once logged-in to the IBM cloud go to catalog and select AI in category and select Watson Assistant
name your service as per your wish and select Dallas as region (it is the closest region to India) and hit on create



III. After selecting Watson Assistant option, window given below will appear.



IV. Once the service is created, select resource list from the three-line menu and select your service (College Admission in my case)



V. From the new page click Launch Watson Assistant and click on create a skill.

❖ Before getting into further we must have some knowledge of basic terminologies of virtual Assistant.

➢ **Skill** (formerly Workspace):

• Sometimes referred to as *dialog skills*.

• Acts as a container for all of the artefacts and training data.

• A skill contains the following types of artefacts:

- Intents
- Entities
- Dialog

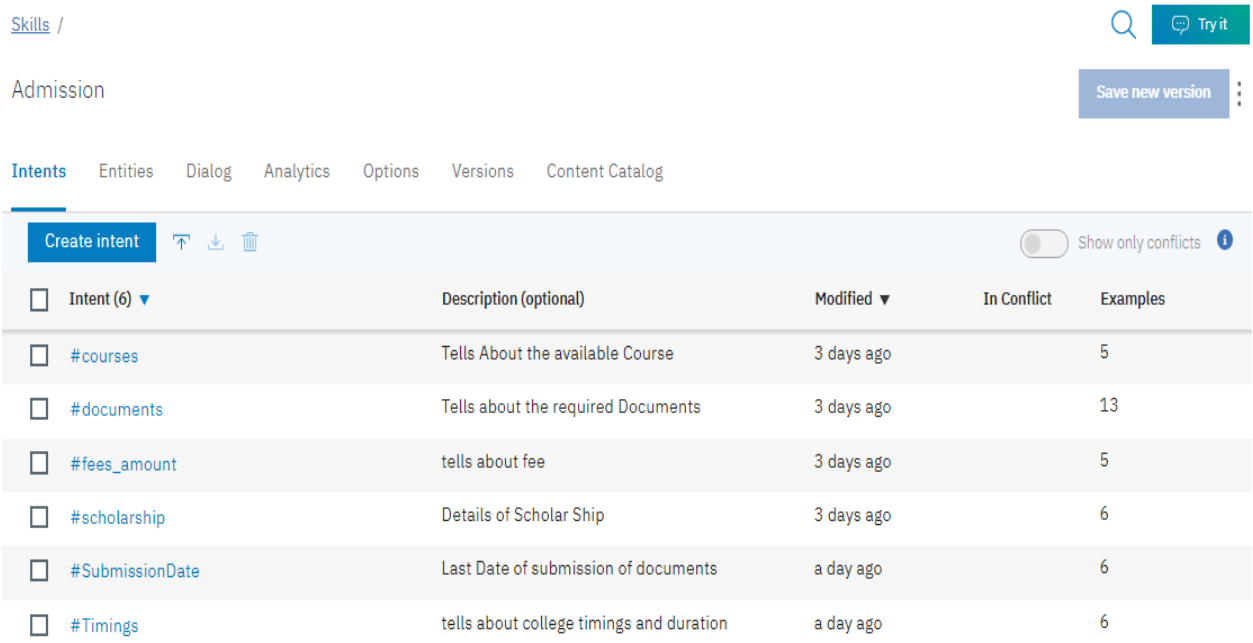
• A skill uses the unique training data to build a machine learning model that can recognize these and similar user inputs. Each time

that you add or change the training data, the training process is triggered to ensure that the underlying model stays up-to-date.

- **Intent:**

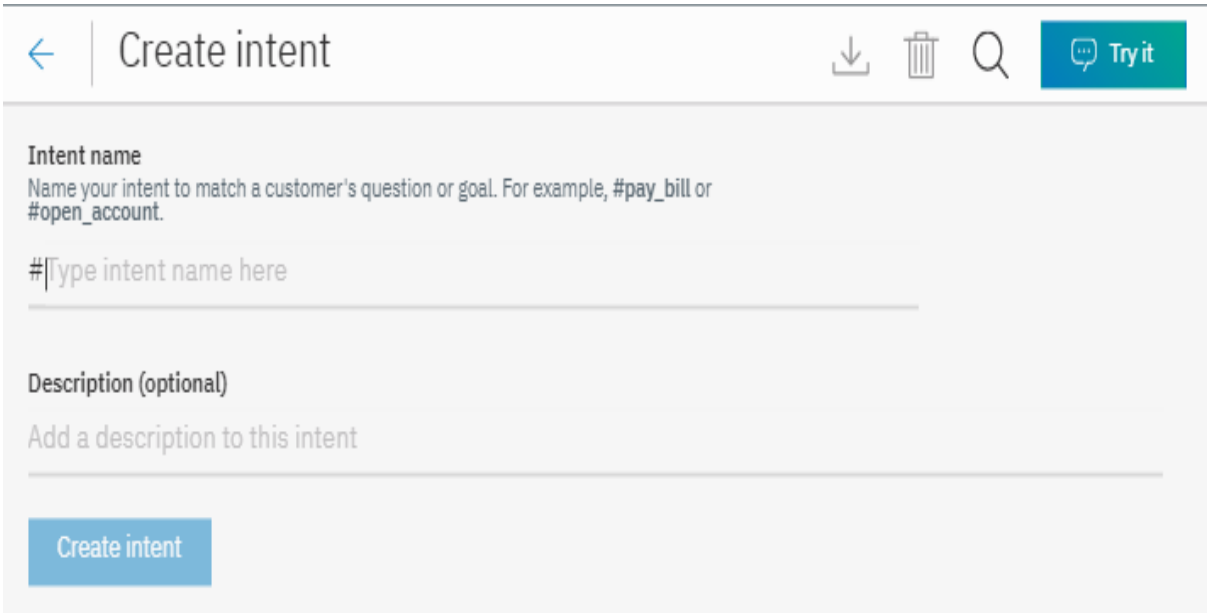
  • Represents the purpose of or goal of a user's input.
  • Tends to be general, such as a question about business location or a bill payment.

VI. Click on create skill.

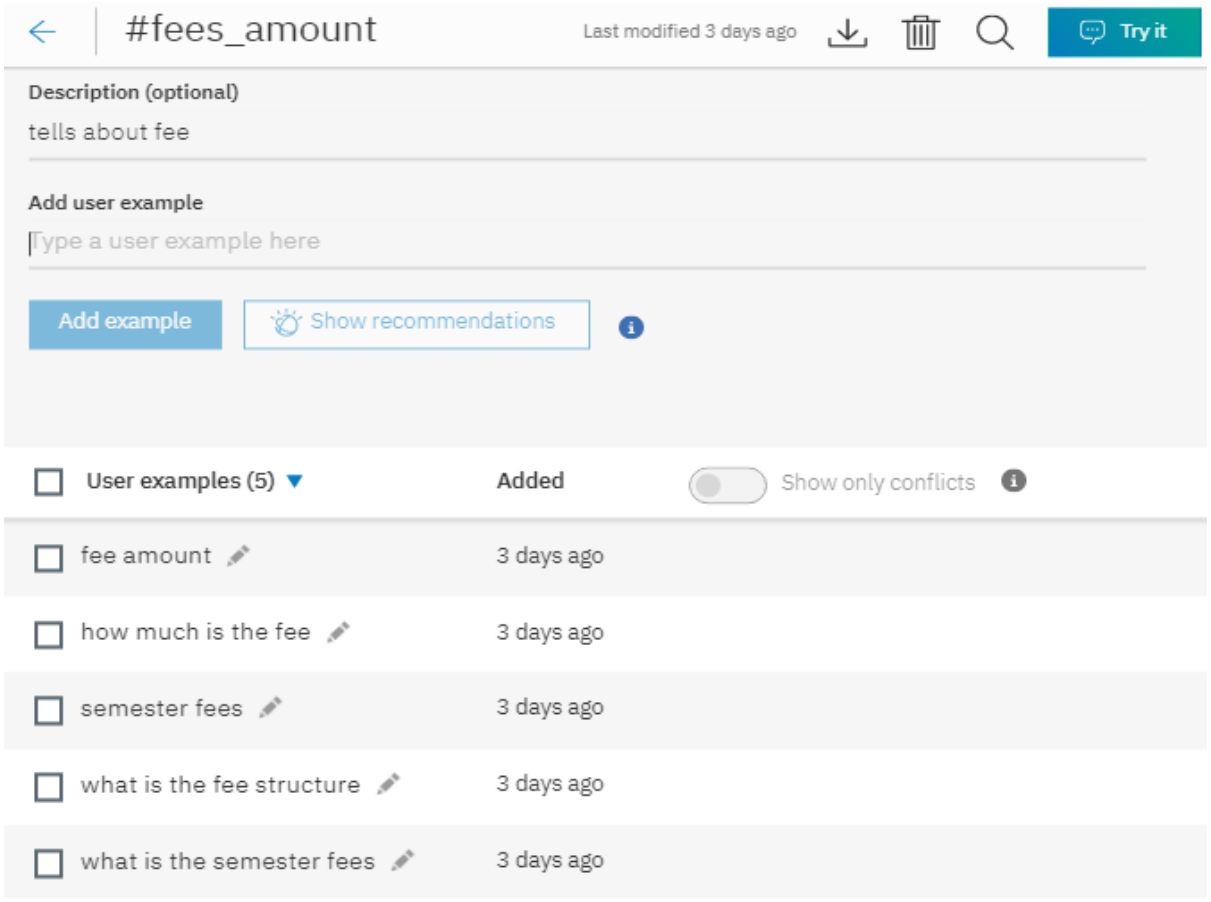


Click on create intent

VII. Click on create intents.
Make as much as possible intents with at least 5 examples.



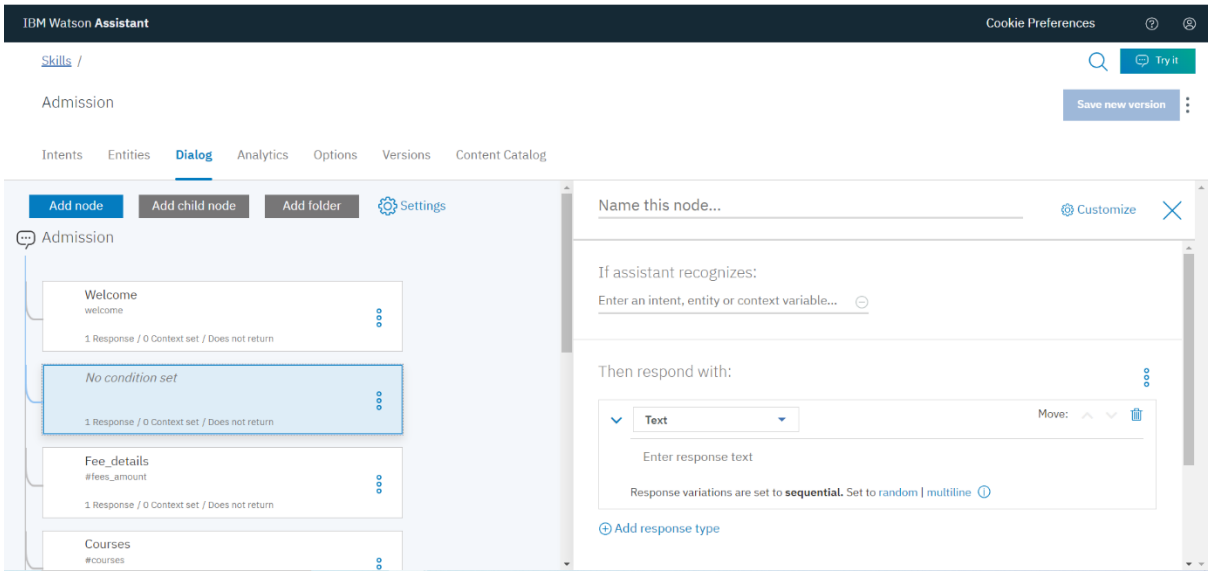
Click on create intent



Create at least 5 examples per intent

VIII. Prepare Dialogs

- Click on dialog and click create dialog.
- Click on the three dots in the front of welcome and click add node below.
- Click on the blank node.



- Name the node as per wish and fill any entity created in if assistant recognizes.
  select response type as text and give as many responses as per requirement.
  we can choose response to be multiline sequential or random.

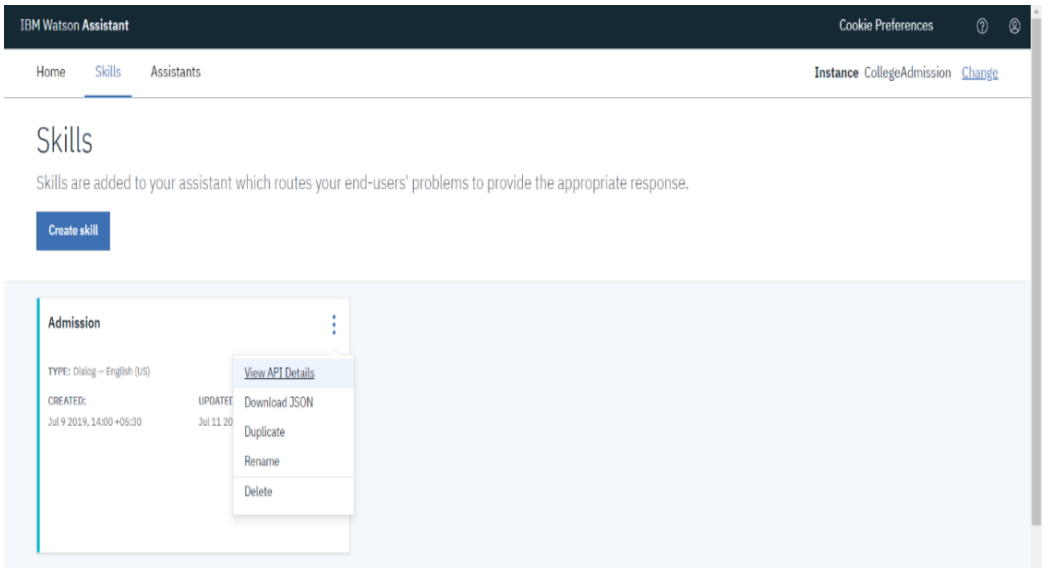
Setting UP responses and Assistant is done now, we may begin with the implementation of bot in python.

2. Prerequisite before implementing in Python:

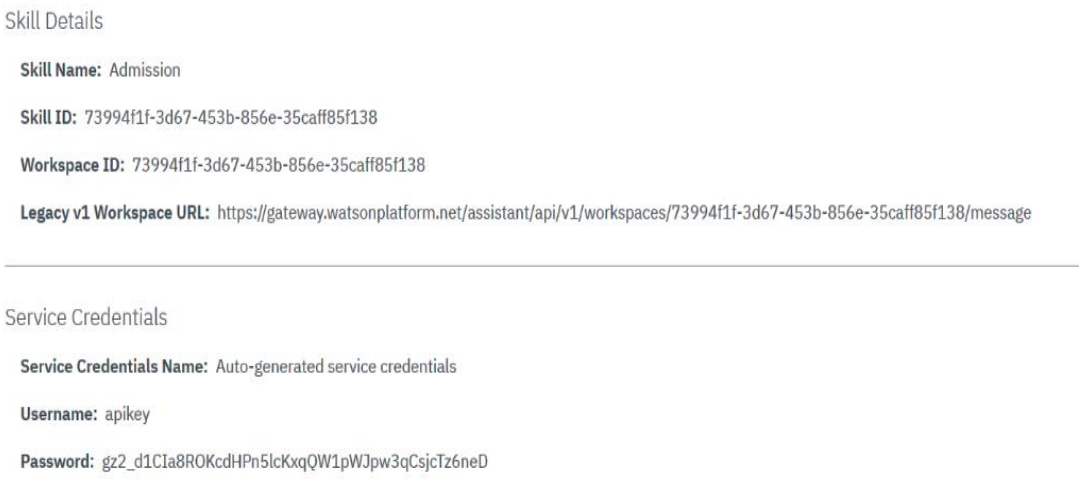
- Before implementing the Assistant on the python, we need following:
  - API Key
  - URL
  - Workspace key

❖ To get API key and URL go to resource list and select assistant, click on show credentials and copy API and URL.

❖ To get workspace key go to skill creation page and click on three dots in front of skill name and click on view API details and copy Workspace id.



Click on View API detail



Copy Workspace ID

3. SSL verification:-

SSL verification is just like logging in to network, to login into SSL we need API key and URL and to link it with skill we need workspace key.

4. Getting output in json format:-

```
import json
import ibm_watson
    assistant = ibm_watson.AssistantV1(
        version='2019-02-28',
        iam_apikey=paste API key here,
        url=paste URL here
    )

    response = assistant.message(
        workspace_id=paste workspace ID here,
        input={
            'text': input('  "enter your query"\t ') #input goes
here

        }
    ).get_result()
    output=json.dumps(response,indent=2)
    print(output)
```

This will give output in json format like this (this does not look good), it give the whole information that happened in cloud

```
{
  "intents": [
    {
```

```
      "intent": "fees_amount", #intent detected
      "confidence": 0.8667376995086671 #intent Accuracy Level
    }
  ],
  "entities": [],
  "input": {
    "text": "fees" #input we entered
  },
  "output": {
    "generic": [
      {
        "response_type": "text",
        "text": "the Admission Fees is rupees 6000, after it the fee
per semester is rupees 2850\nGirls have less semester fee Amount of
rupees 1450"
      }#response
    ],
    "text": [
      "the Admission Fees is rupees 6000, after it the fee per
semester is rupees 2850",
      "Girls have less semester fee Amount of rupees 1450"
    ], #we only need this list, below this is completely useless for
us
    "nodes_visited": [
      "node_1_1562661491055"
    ],
    "log_messages": []
  },
  "context": {
    "conversation_id": "5bd733a9-03d4-4a47-a358-129cf8b9bc04",
    "system": {
      "initialized": true,
      "dialog_stack": [
        {
          "dialog_node": "root"
        }
      ],
      "dialog_turn_counter": 1,
     "dialog_request_counter": 1,
      "branch_exited": true,
      "branch_exited_reason": "completed"
    }
  }
}
```

But we only need only output so we have to index the output, we can do indexing as following:

```
    output=json.dumps(response['output']['text'],
indent=2).split('[')[1]
    output=output.split(']')[0]
    output=str(output)
    print("\n"+output)
```

We can also use split function to remove the square brackets ([ ]) from output
Now we know how to give input and take output from assistant in python

5. Now here is a complete code with output using python:-

```
import json
import ibm_watson
print('  "Hi I am an virtual assistant to help your queries regarding
admission"\n  "I run on IBM cloud"\n  "I am made my
Neetu,Gautam,Mayanak,Riya and Aadarsh"\n  ')
while True:
    assistant = ibm_watson.AssistantV1(
        version='2019-02-28',

iam_apikey='gz2_d1CIa8ROKcdHPn5lcKxqQW1pWJpw3qCsjcTz6neD',
        url='https://gateway.watsonplatform.net/assistant/api'
    )

    response = assistant.message(
        workspace_id='73994f1f-3d67-453b-856e-35caff85f138',
        input={
            'text': input('  "enter your query"\t ')
        }
    ).get_result()
    output=json.dumps(response['output']['text'], indent=2).split('[')[1]
    output=output.split(']')[0]
    output=str(output)
    print("\n"+output)
    a=str(input('  "do you have any other query(yes/no)"\t '))
    a.lower
    condition=a
    if condition=='yes':
        pass
    elif condition=='no':
        print('  "thanks for using me"\n  "have a nice"\n ')
```

```
        break
    else:
        print('  "invalid input"\n  "stopping service"\n ')
```

Output:-

```
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.4.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Mohit/Desktop/AssistantTextIO.py',
wdir='C:/Users/Mohit/Desktop')
  "Hi I am an virtual assistant to help your queries regarding admission"
  "I run on IBM cloud"
  "I am made my Neetu,Gautam,Mayanak,Riya and Aadarsh"

  "enter your query"      how many courses are there
  "we have courses of Computer Engineering , civil Engineering, Mechanical
Engineering, Automobile Engineering, Electrical Engineering, Electronic
Engineering.",
  "Courses are allotted to students according to merit bases, if the student is
unhappy with allotment the he or she may prefer for Open Counselling "

  "do you have any other query(yes/no)"    no
  "thanks for using me"
  "have a nice"

In [2]: runfile('C:/Users/Mohit/Desktop/AssistantTextIO.py',
wdir='C:/Users/Mohit/Desktop')
  "Hi I am an virtual assistant to help your queries regarding admission"
  "I run on IBM cloud"
  "I am made my Neetu,Gautam,Mayanak,Riya and Aadarsh"

  "enter your query"      admission fees for girl

  "the Admission Fees is rupees 6000, after it the fee per semester is rupees
2850",
  "Girls have less semester fee Amount of rupees 1450"
```

"do you have any other query(yes/no)" yes
"enter your query" documents required

"applicants are required to submit two set of following self-attested documents",
"Birth Proof",
"Domicile also known as Residence certificate",
"aadhar card",
"10th mark sheet 12th mark sheet if filling form on 12th base ",
"proof of filling counselling fees",
"2 photos"

"do you have any other query(yes/no)" yes

"enter your query" scholarship

"sorry I couldn't catch you, you may try following inputs",

"fees",
"courses",
"documents",
"scholarship",
"timings",
"last date of documents submission"

"do you have any other query(yes/no)" yes

"enter your query" timing

"college timing is 9:00 to 4:00",
"lunch timing 1:00 to 2:00",
"college Days Monday to Friday",
"weekly holidays Saturday and Sunday"

"do you have any other query(yes/no)" no

"thanks for using me"
"have a nice"

Here we entered text Input and got text Output, Now we are going to make a chatbot to get voice output using voice input.

Chatbot with voice I/O

Libraries required:-

1. pyaudio
2. playsound
3. speech_recognition
4. GTTS
5. json
6. ibm-watson

For speech input we imported speech_recognition, it recognizes the voice, and uses Google services to recognize voice and convert it into text.

Code for our project to listen audio:-

```
import speech_recognition as sr #importing Library
r=sr.Recognizer() #storing recognizer class into an object
with sr.Microphone() as source: # setting up microphone as listning source
    askquery=str('  "ask your query"\t ') #Message to display before voice input output
    print(askquery)
    r.adjust_for_ambient_noise(source) #for noise cancellation
    audio=r.listen(source,phrase_time_limit=5) #listning audio
try:
    query=r.recognize_google(audio) # sending audio to google and recognize it
    print("\t"+query)
except:
    pass
```

We can remove try and except block but this block allows us error handling if we discover any type of error while using it so we are going to use it
output:-

```
"ask your query"
      mike testing mike testing 123
```

Text to audio conversion:-

For text to audio conversion we need GTTS from gtts , it stands for Google text to speech and it uses Google services to convert text into Audio .

To listen audio, we also need play sound library, we need to use text to voice conversion a number of times, so will make a function for this.

```
from gtts import gTTS
import os
from playsound import playsound

def voice(inpt):
    myobj = gTTS(text=inpt, lang="en", slow=False)
    myobj.save("VoiceOut.mp3")
    playsound('VoiceOut.mp3')
    os.remove('VoiceOut.mp3')
```

As the gTTS stores the audio in mp3 file format it's easy to listen it, but if we will use this code again an error will occur saying VoiceOut.mp3 permission denied, so in order to overcome this, we have to delete VoiceOut.mp.
For this we have to import os library and use os.remove('file name') function

Now we will link all these functions using function calls and make it struck into an infinite loop and make a condition to break it as per user need.

Our whole code is:

```
import os
import json
import ibm_watson
from gtts import gTTS
import speech_recognition as sr
from playsound import playsound
def voice(input):
    myobj = gTTS(text=inpt, lang="en", slow=False)
    myobj.save("VoiceOut.mp3")
    playsound('VoiceOut.mp3')
    os.remove('VoiceOut.mp3')

welcome=str('  "Hi I am an virtual assistant to help your queries regarding
admission"\n  "I run on IBM cloud"\n  "I am made by Neetu,Mayank,Gautam,Riya
and Aaadrsh"\n')
print(welcome)
voice(welcome)
while True:
    r=sr.Recognizer()
    with sr.Microphone() as source:
```

```
        askquery=str('  "ask your query"\t ')
        voice(askquery)
        print(askquery)
        r.adjust_for_ambient_noise(source)
        audio=r.listen(source,phrase_time_limit=5)
    try:
        query=r.recognize_google(audio)
        print("\t"+query)
    except:
        pass
    if query=='quit':
        qt=str('  "thanks for using me have a nice day"  ')
        voice(qt)
        print(qt)

        break
    elif query=='exit':
        qt=str('  "thanks for using me have a nice day"  ')
        voice(qt)
        print(qt)
        break
    else:
        pass
    assistant = ibm_watson.AssistantV1(
        version='2019-02-28',
        iam_apikey='gz2_d1CIa8ROKcdHPn5lcKxqQW1pWJpw3qCsjcTz6neD',
        url='https://gateway.watsonplatform.net/assistant/api'
        )
    response = assistant.message(
            workspace_id='73994f1f-3d67-453b-856e-35caff85f138',
            input={
                'text': query
            }
        ).get_result()
    output=json.dumps(response['output']['text'], indent=2).split('[')[1]
    output=output.split(']')[0]
    output=str(output)
    print(output)
    voice(output)
    qt=str('  "ask another query or say quit or exit to quit the program on
next step"\n  ')
    print(qt)
    voice(qt)
```

Output: -

```
"Hi I am an virtual assistant to help your queries regarding admission"
"I run on IBM cloud"
"I am made by Gautam,Mayank,Neetu,Riya and Aadrsh"

"ask your query"
      fees structure

"the Admission Fees is rupees 6000, after it the fee per semester is rupees 2850",
"Girls have less semester fee Amount of rupees 1450"

"ask another query or say quit or exit to quit the program on next step"

"ask your query"
      last date of document submission

"the last date of submitting document is  12th august2019"

"ask another query or say quit or exit to quit the program on next step"

"ask your query"
      exit
"thanks for using me have a nice day"
```

The statements in quotes are the outputs from the assistant and sentences without quotes are input.

GUI with voice input and voice plus text output: -

Libraries required

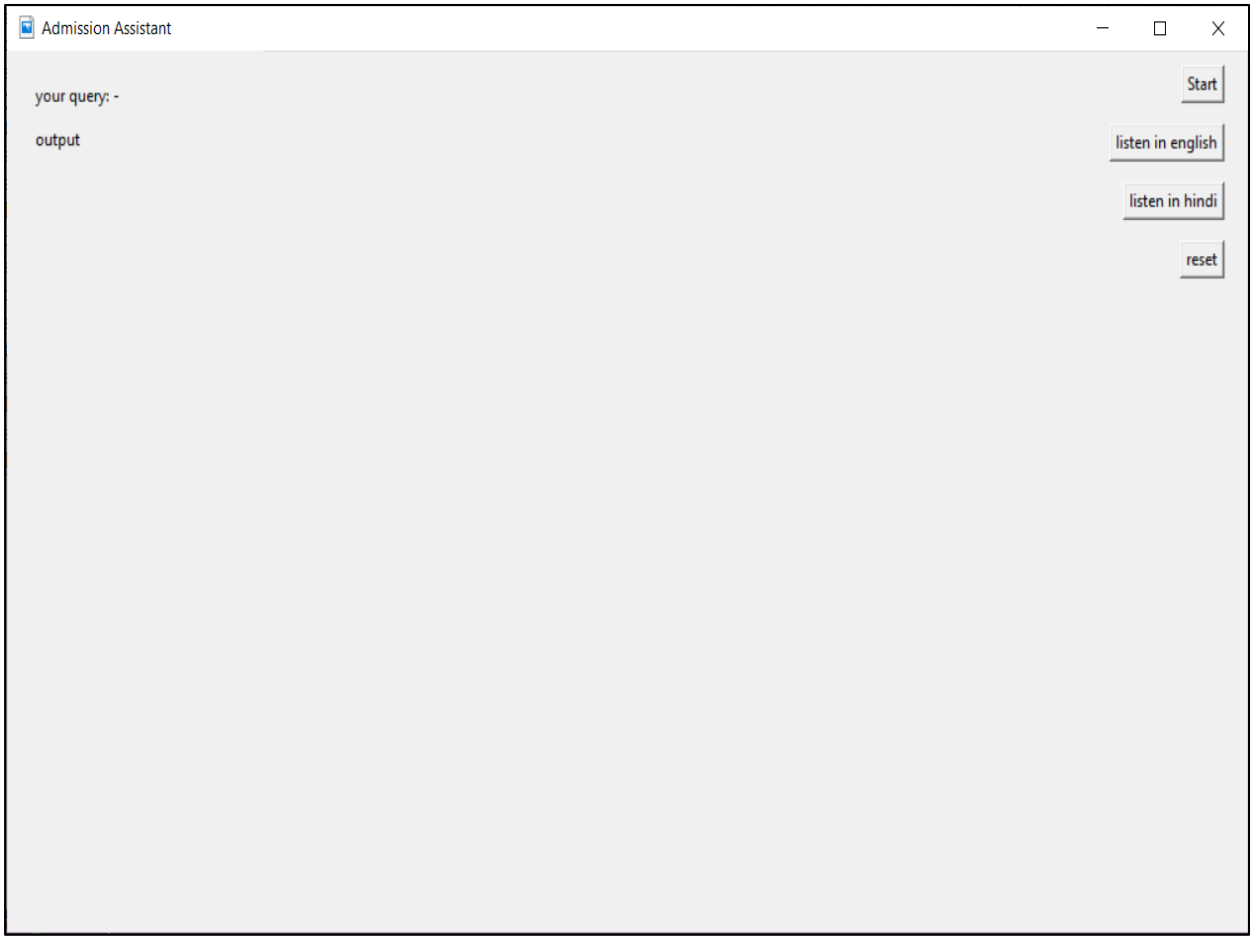
1. pyaudio
2. playsound
3. speech_recognition
4. GTTS
5. json
6. ibm-watson
7. tkinter

GUI works on the basis of function calling, for GUI we first need to decide the position of buttons, labels and text.

Code to make a GUI window:-

```
from tkinter import *
root = Tk() #assigning Tk class to object
root.title("Admission Assistant") #tittle of our wondow
root.iconbitmap('logo.jpg') #icon for tittle bar
cvs=Canvas(root,height=600,width=1000) #it's like a normal canvas where we use
to draw
cvs.pack() #packing canvas to root
can_in=Button(cvs,text='Start') #start button
can_in.place(x=950,y=10) #placing it on canvas
can_out=Label(cvs,text='output') #output label
can_out.place(x=20,y=50) #placing it on canvas
lien=Button(cvs,text='listen in English') #listen in English button
lien.place(x=892,y=50) #placing it on canvas
lihi=Button(cvs,text='listen in Hindi') #listen in Hindi button
lihi.place(x=903,y=90) #placing it on canvas
resetb=Button(cvs,text='reset') #reset Button
resetb.place(x=949,y=130) #placing it on canvas
voin=Label(cvs,text='your query: -') #query Label
voin.place(x=20,y=20) #placing it on canvas
root.mainloop() ''' used to keep window in loop so that we can see the window
clearly, else whole process will get completed in fraction of second and we
will not be able to work on GUI'''
```

After executing this code, window given below will open.



Next work is to make functions that should be assigned to buttons.

1. For Start button :-

For start button we wanted a function that should start listening from a microphone and give voice output so we declared a function sreco() for voice input and CODE-C for voice output.

```
def sreco():
    r=sr.Recognizer()
    with sr.Microphone() as source:
        askquery=str('  "ask your query"\t ')
        voice(askquery)
        print(askquery)
        r.adjust_for_ambient_noise(source)
        audio=r.listen(source,phrase_time_limit=5)
    try:
        query=r.recognize_google(audio)
        global voin
        voin=Label(cvs,text='your query: -\t'+query) #query label
```

```
        voin.place(x=20,y=20)
        print("\t"+query)
    except:
        pass
    watson(query)
```

We have place query label (voin) here to show the input, we've declared it as global for future modifications, and also we declare all the labels and buttons as global for future modifications to stay on safe side.

Now we will declare Watson function that will be called by sreco(), sreco() will pass the query to Watson and Watson will give the output in form of text, here we will destroy the start button in order to prevent overlapping of multiple input and output.

We will also place reset, listen in Hindi and listen in English buttons.

Our Watson function is as follow:-

```
def watson(query):
    assistant = ibm_watson.AssistantV1(
    version='2019-02-28',
    iam_apikey='gz2_d1CIa8RcdHPn5lcKxqQW1pWJpw3qCsjcTz6neD',
    url='https://gateway.watsonplatform.net/assistant/api'
    )

    response = assistant.message(
            workspace_id='73994f1f-3d67-453b-856e-35caff85f138',
            input={
                'text': query
            }
        ).get_result()
    global output
    output=json.dumps(response['output']['text'], indent=2).split('[')[1]
    output=output.split(']')[0]
    output=str(output)
    print(output)
    global lihi , lien , can_out , resetb
    can_in.destroy() #distroying start button
    can_out=Label(cvs,text=output) #output labe;
    can_out.place(x=20,y=50) #placing it on canvas
    lien=Button(cvs,text='listen in English',command=voen) #listen in hindi
button
    lien.place(x=892,y=50) #placing it on canvas
    lihi=Button(cvs,text='listen in Hindi',command=vohi) #listen in hindi
button
    lihi.place(x=903,y=90) #placing it on canvas
    resetb=Button(cvs,text='reset',command=reset) #reset button
```

```
    resetb.place(x=949,y=130) #placing it on canvas
```

2. For listen in English:
   For this button we made a simple function that will call the voice function and will pass output to that function we named in voen().

```
def voen():
    voice(output)
```

3. For listen in Hindi:
   For this button we declared a function that will translate our output to Hindi and will pass it to our voice function vohi().

```
def vohi():
    translator= Translator(from_lang='en',to_lang="hindi") #initializing
translator
    hindi = translator.translate(output) #translating output
    voice(hindi)
```

4. To reset our application we need to destroy 'listen in Hindi' , 'listen in English' , 'reset' buttons an regenerate 'start' button, we also need to destroy 'query' and 'output' label we.
   Now we need to declare start button as global. Our reset functions as follows:

```
def reset():
    voin.destroy()
    lihi.destroy()
    lien.destroy()
    can_out.destroy()
    resetb.destroy()
    global can_in
    can_in=Button(cvs,text='Start',command=sreco)
    can_in.place(x=950,y=10)
```

Now, we just have to place our all functions in correct order, and our program is ready to go.

Complete code of my program is as follows:

```
import os
import json
import ibm_watson
from gtts import gTTS
import speech_recognition as sr
from playsound import playsound
from tkinter import *
from translate import Translator

def voice(inpt):
    myobj = gTTS(text=inpt, lang="en", slow=False)
    myobj.save("VoiceOut.mp3")
    playsound('VoiceOut.mp3')
    os.remove('VoiceOut.mp3')

def sreco():
    r=sr.Recognizer()
    with sr.Microphone() as source:
        askquery=str('  "ask your query"\t ')
        voice(askquery)
        print(askquery)
        r.adjust_for_ambient_noise(source)
        audio=r.listen(source,phrase_time_limit=5)
    try:
        query=r.recognize_google(audio)
        global voin
        voin=Label(cvs,text='your query: -\t'+query) #query label
        voin.place(x=20,y=20)
        print("\t"+query)
    except:
        pass
    watson(query)

def watson(query):
    assistant = ibm_watson.AssistantV1(
    version='2019-02-28',
    iam_apikey='gz2_d1CIa8ROKcdHPn5lcKxqQW1pWJpw3qCsjcTz6neD',
    url='https://gateway.watsonplatform.net/assistant/api'
    )

    response = assistant.message(
            workspace_id='73994f1f-3d67-453b-856e-35caff85f138',
            input={
                'text': query
            }
        ).get_result()
    global output
```

```
    output=json.dumps(response['output']['text'], indent=2).split('[')[1]
    output=output.split(']')[0]
    output=str(output)
    print(output)
    global lihi , lien , can_out , resetb
    can_in.destroy() #distroying start button
    can_out=Label(cvs,text=output) #output labe;
    can_out.place(x=20,y=50) #placing it on canvas
    lien=Button(cvs,text='listen in English',command=voen) #listen in hindi
button
    lien.place(x=892,y=50) #placing it on canvas
    lihi=Button(cvs,text='listen in Hindi',command=vohi) #listen in hindi
button
    lihi.place(x=903,y=90) #placing it on canvas
    resetb=Button(cvs,text='reset',command=reset) #reset button
    resetb.place(x=949,y=130) #placing it on canvas

def voen():
    voice(output)

def vohi():
    translator= Translator(from_lang='en',to_lang="hindi") #initializing
translator
    hindi = translator.translate(output) #translating output
    voice(hindi)

def reset():
    voin.destroy()
    lihi.destroy()
    lien.destroy()
    can_out.destroy()
    resetb.destroy()
    global can_in
    can_in=Button(cvs,text='Start',command=sreco)
    can_in.place(x=950,y=10)

root = Tk()
root.title("Admission Assistant")
root.iconbitmap('logo.jpg')
cvs=Canvas(root,height=600,width=1000)
cvs.pack()
can_in=Button(cvs,text='Start',command=sreco)
can_in.place(x=950,y=10)
root.mainloop()
```

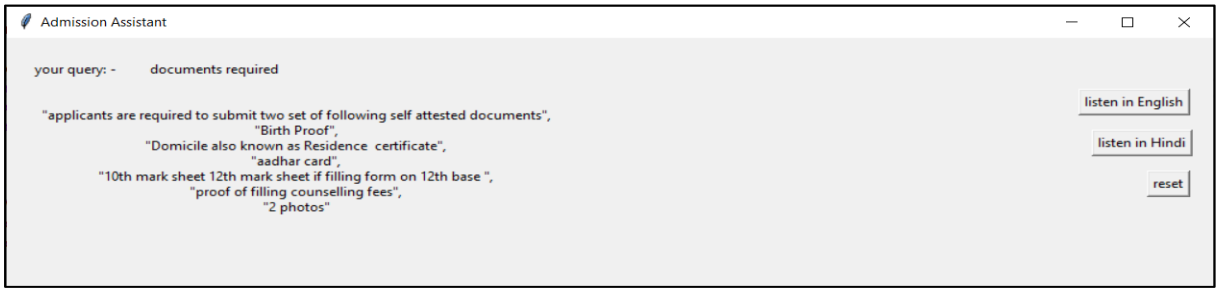
Below are some output images of our GUI program: -

At start

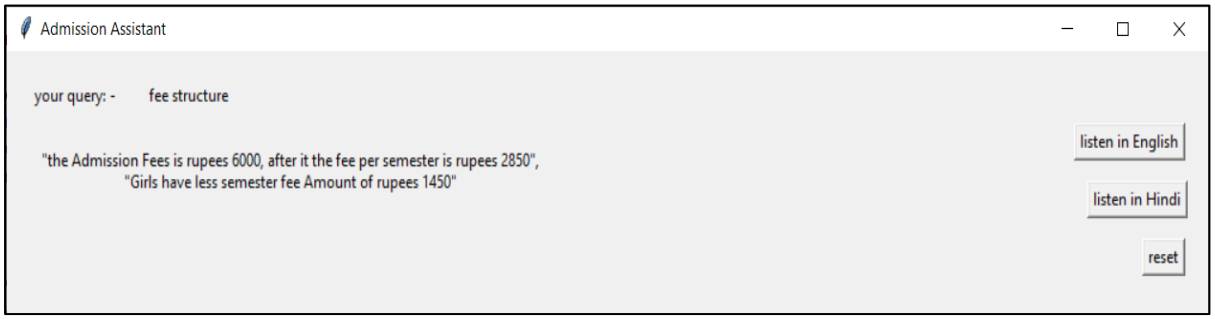
This will be the first screen to be appear having start button. After we click on 'start' new screen having all GUI buttons will appear:-

All GUI buttons

Now all we have to do is to give input in voice form and we will get output in voice plus text form:-

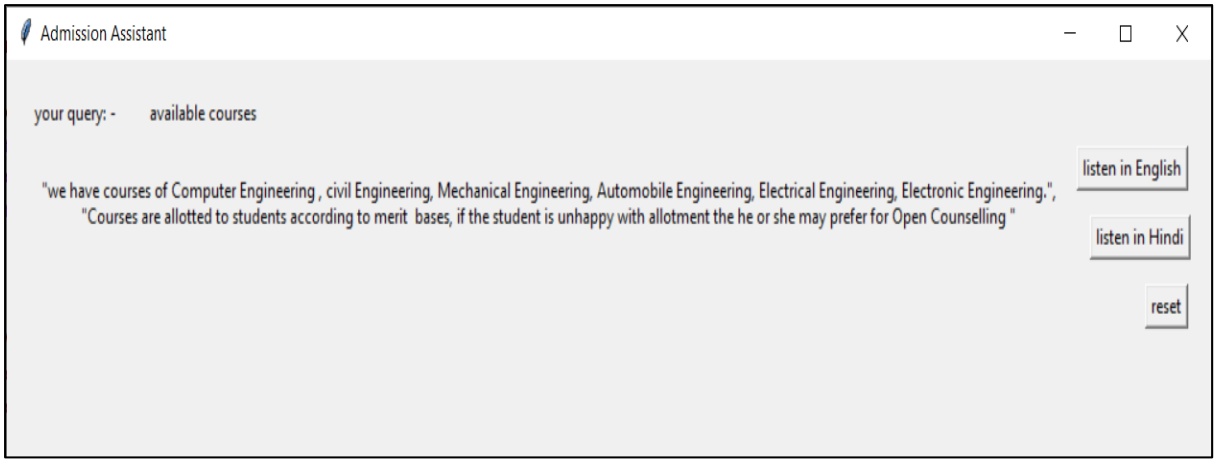


Query 1



Query 2

Query 3



Query 4

When we click on listen in Hindi, or listen in English button, we will get voice output in the language we have selected.

Reset

After pressing reset button we will come back to our first window, now we can close our chatbot.

Importance of chatbot:-

It will help and enable chatbot to converse with both via their voice natural language or with textual data. The voice chatbot then answer back using pre-record message of both i.e. message canl be both listened and displayed.