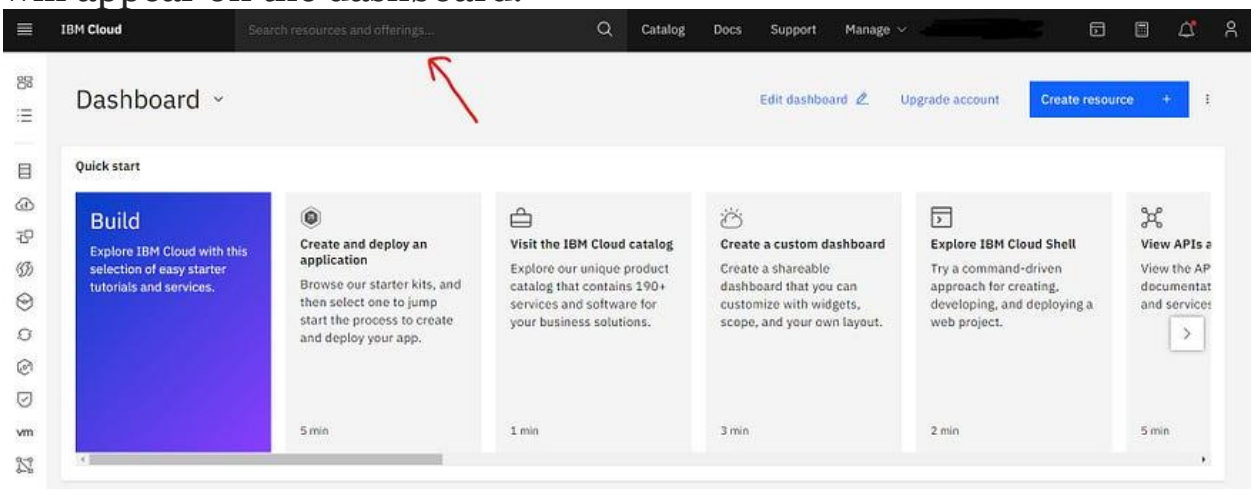
Building the chatbot

we need to have an IBM Cloud account to use the IBM Watson Assistant service. Sign in to our IBM Cloud account. In case we don't have one, we can sign up for free here.

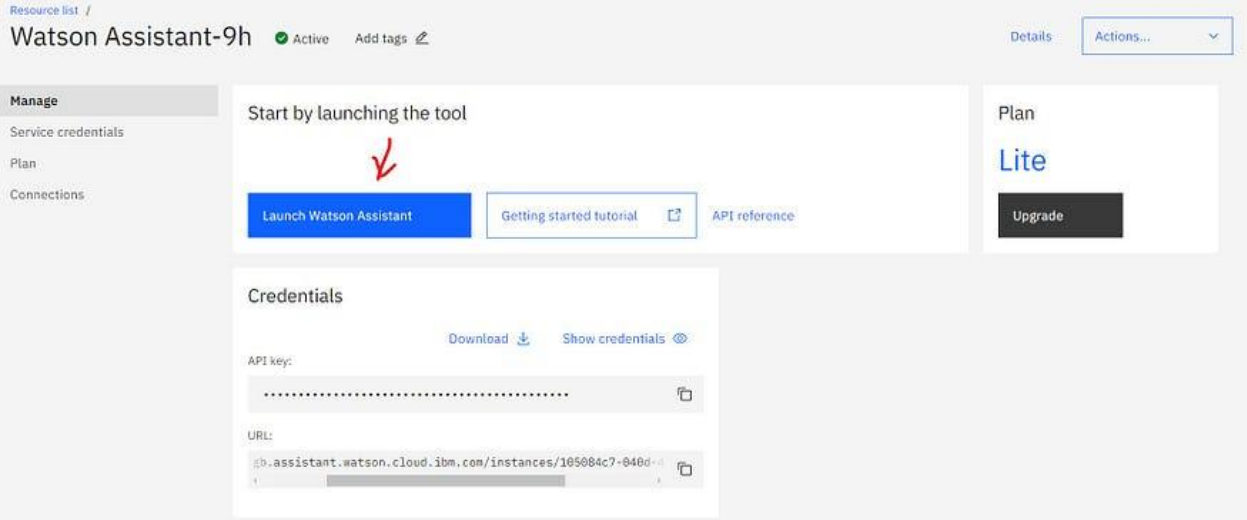
Step 1: Creating an instance of Watson Assistant Service

Once you have signed in to your IBM Cloud Account, the below screen will appear on the dashboard.



Type " Watson Assistant" in the search bar and create a service of Watson Assistant.

Once you have created the service of Watson Assistant, click on it and the following screen will appear.

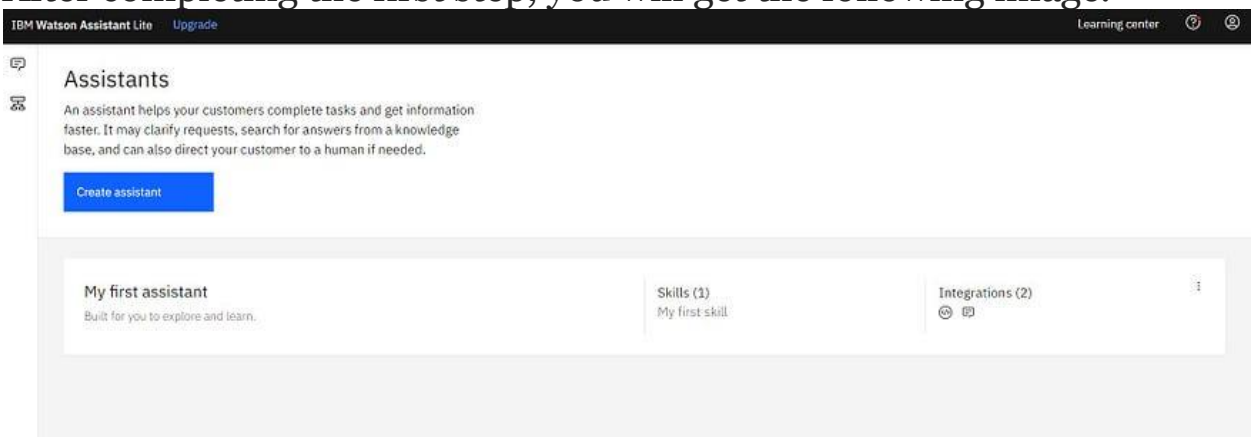


Click on “Launch Watson Assistant”

Now click on “Launch Watson Assistant” to launch the service.

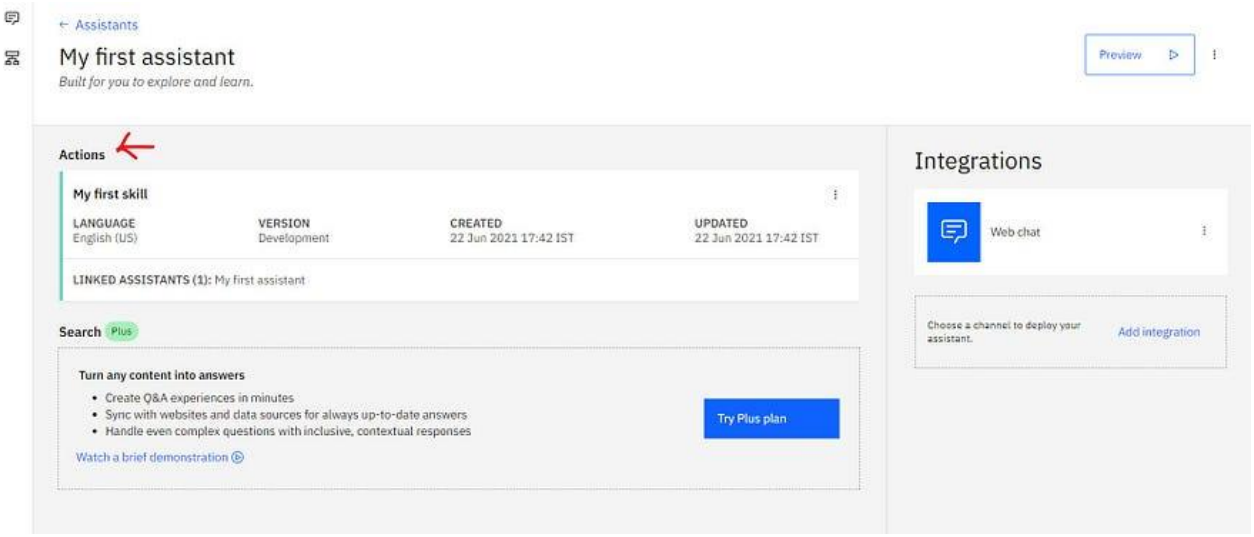
Step 2: Add a Dialog skill to the Assistant

After completing the first step, you will get the following image.



In this step, we will add the Dialog skill to our assistant. Dialog skills will help our assistant to perform the mentioned task.

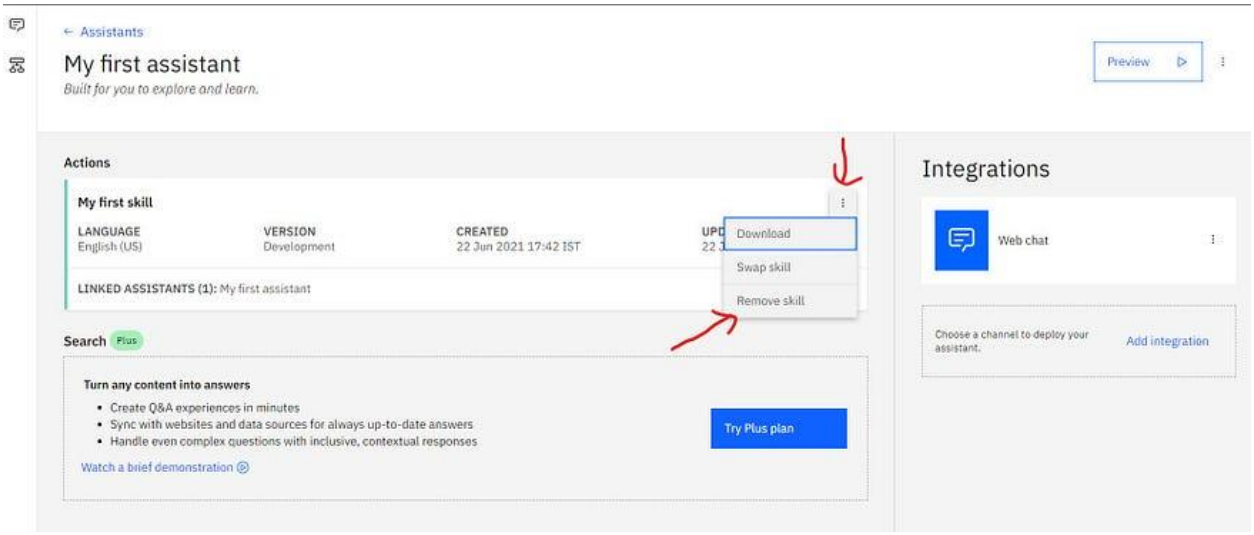
To add a dialog skill to the assistant, click on the “ My first assistant”. You will get the following image.



Action skill linked to “ My first assistant”

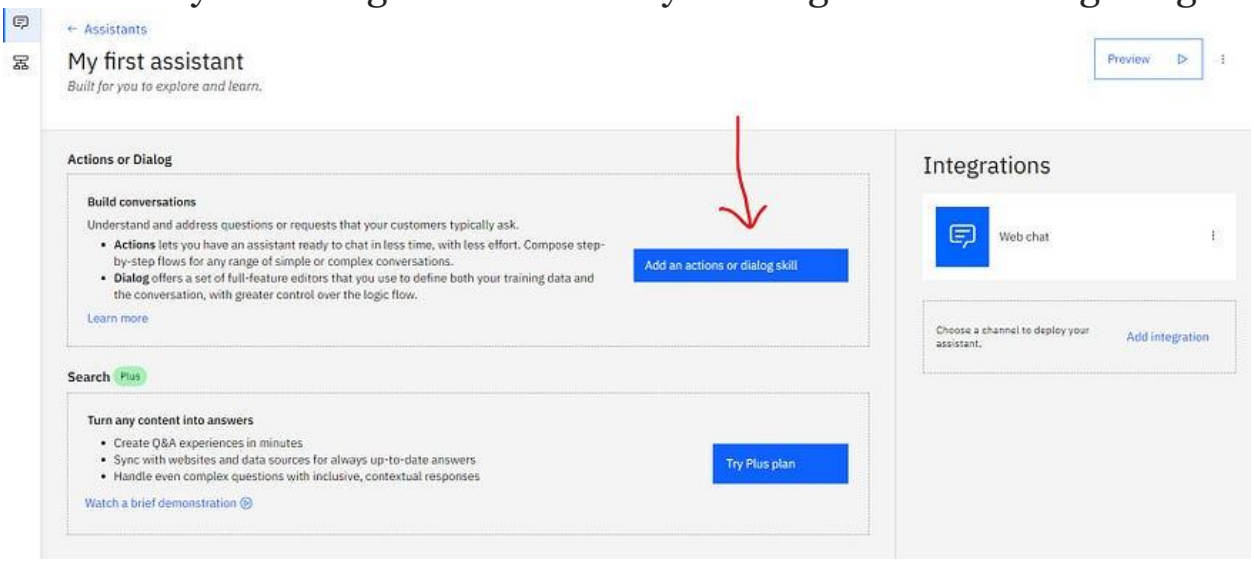
Here you can see, there is an actions skill linked to the assistant. In some cases, there is an action skill linked to the assistant and in others, there is a dialog skill linked to the assistant.

If there is a dialog skill linked to the assistant then we don’t need to change anything but in our case, there is an action skill linked to the assistant & we want to link the dialog skill. To do so, first, we remove the action skill from the assistant.



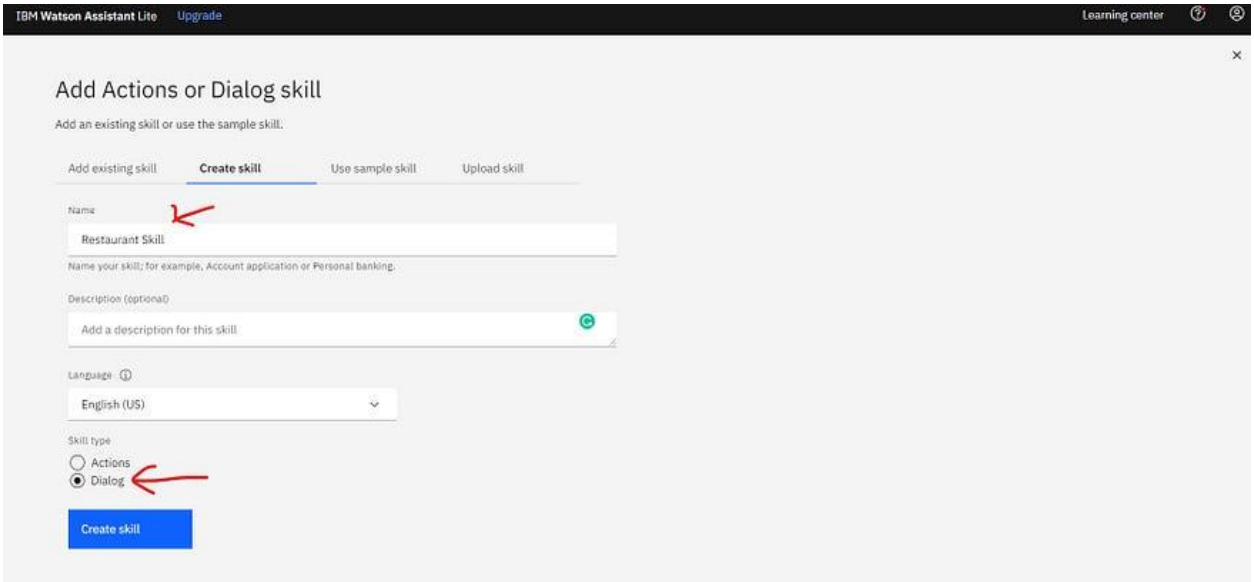
Removing an actions skill

To remove an existing action skill, click on the three dots in the upper right corner of *My first skill* and click on the *remove skill*. After successfully removing the action skill you will get the following image.



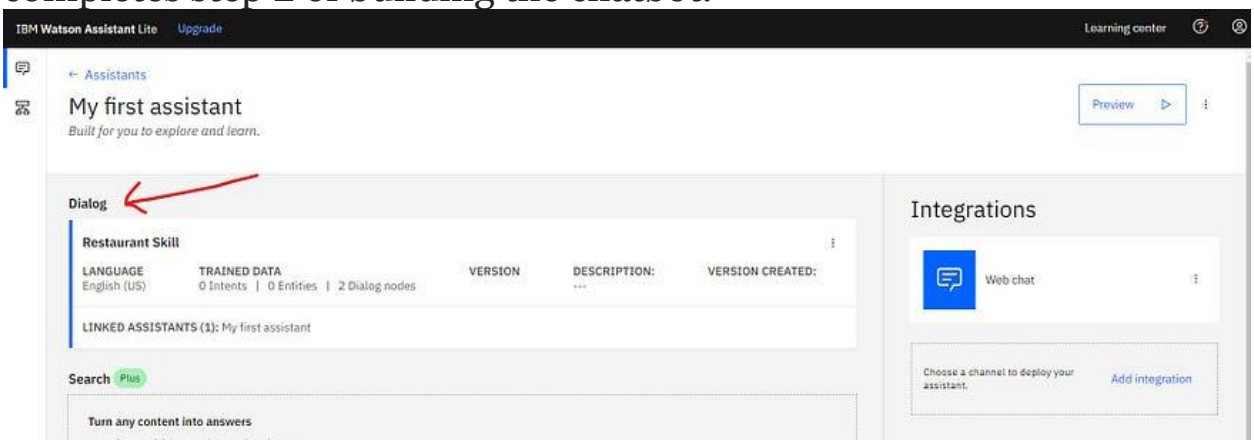
Add a dialog skill

Now to add a dialog skill, click on “Add an action or dialog skill” and Create a new dialog skill with the name “ Restaurant Skill” and click on the *Create skill*.



Creating a new Dialog skill.

After completing this, a new dialog skill is added to our assistant. This completes step 2 of building the chatbot.



Assistant with the Dialog Skill.

Step 3: Create an Intent and Add to the Skill

In this step, we will create an Intent and add it to the skill. But before that, we need to understand what is an intent?

According to Watson Assistant

An intent is a collection of user statements that have the same meaning. By creating intents, you train your assistant to understand the variety of ways users express a goal.

An intent is simply a collection of user statements that have the same meaning. These statements can be the statement from which users want to express a goal, In our case, something like *Make a booking* for the table.

What are the different ways a user can say to make a booking for a table in the restaurant?

Make a booking for a table.

Reserve a table for me.

Make a reservation for a table in the restaurant.

Book a table.

I want to book a table for dinner.

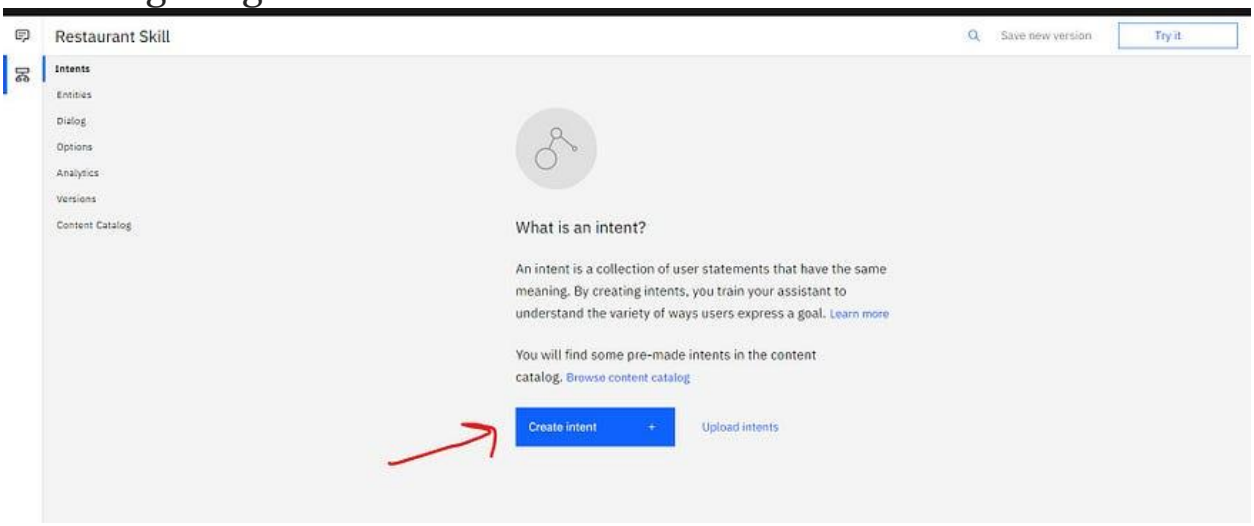
All these examples show the intention of the user to book a table. By looking at these examples, we can easily say that the user wants to book a table. The same goes for the *Intent* in the chatbot.

Intent helps the chatbot to identify the goals that users want to accomplish. These goals can be to make a booking, getting information about operating hours, etc.

In our chatbot, We want to accomplish two main goals: making a booking and getting information about hours of operation. So we will create two intents, one for each.

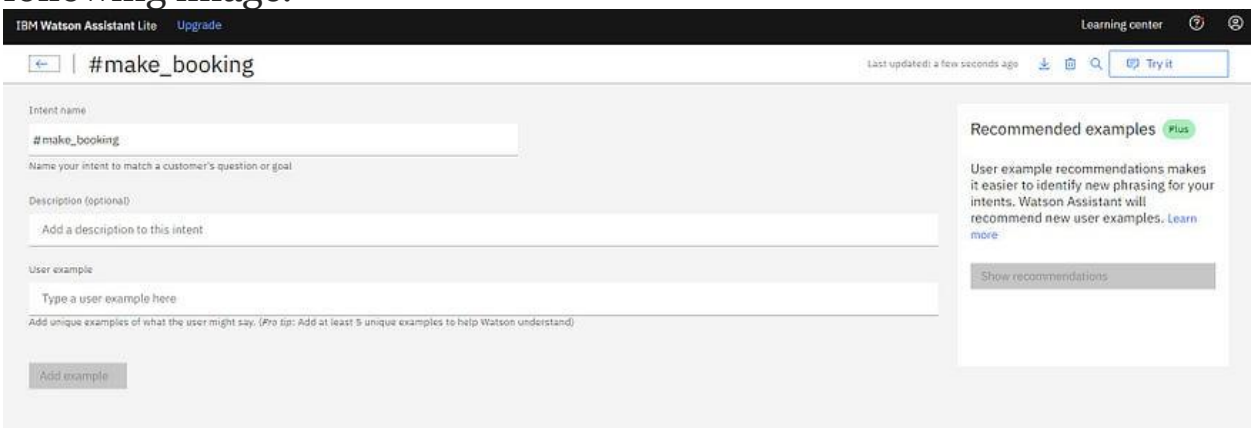
First, We create an intent to understand the user's intention to make a booking for a table.

To create an intent, first, click on the *Restaurant skill*. You will get the following image.



Create an Intent

Now click on the *Create Intent*. After that, type the name of the intent " make_booking" and then click on the create intent. You will get the following image.



#make_booking intent created

Now we need to train the intent by giving at least 5 different examples in which a user can say to a chatbot to make a booking for the table. The examples can be the following as we have seen earlier.

Make a booking for a table.

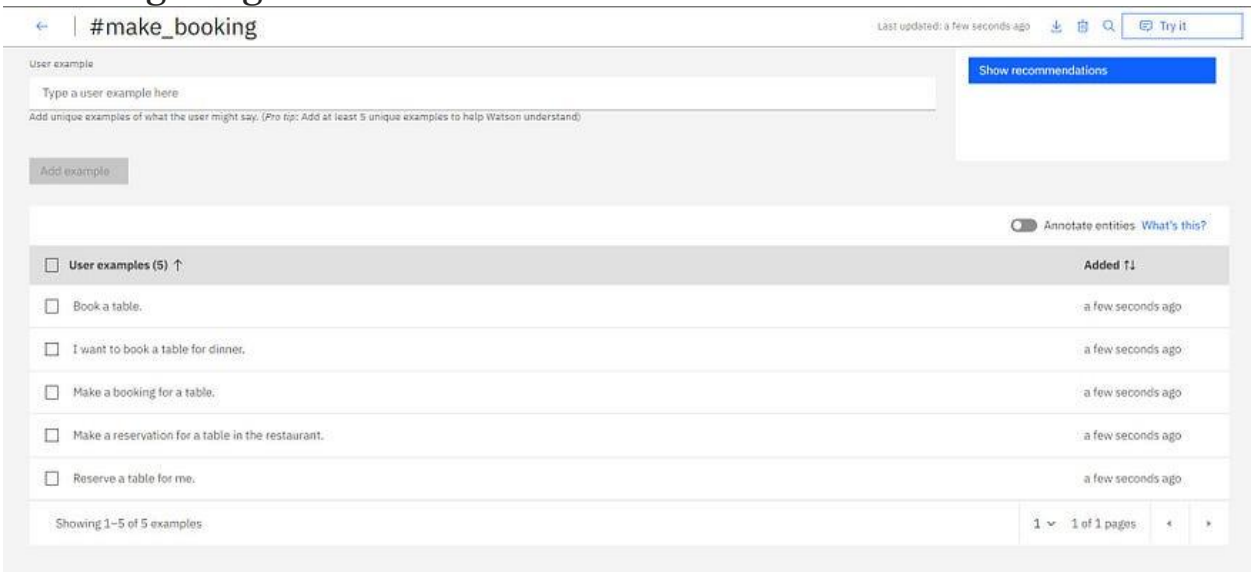
Reserve a table for me.

Make a reservation for a table in the restaurant.

Book a table.

I want to book a table for dinner.

To add examples, just type the statement and then click on *add example*. Once you successfully added all the examples, you will get the following image.



5 different examples added to the intent

Similarly, Create another intent for the *hours of operation* with the following examples.

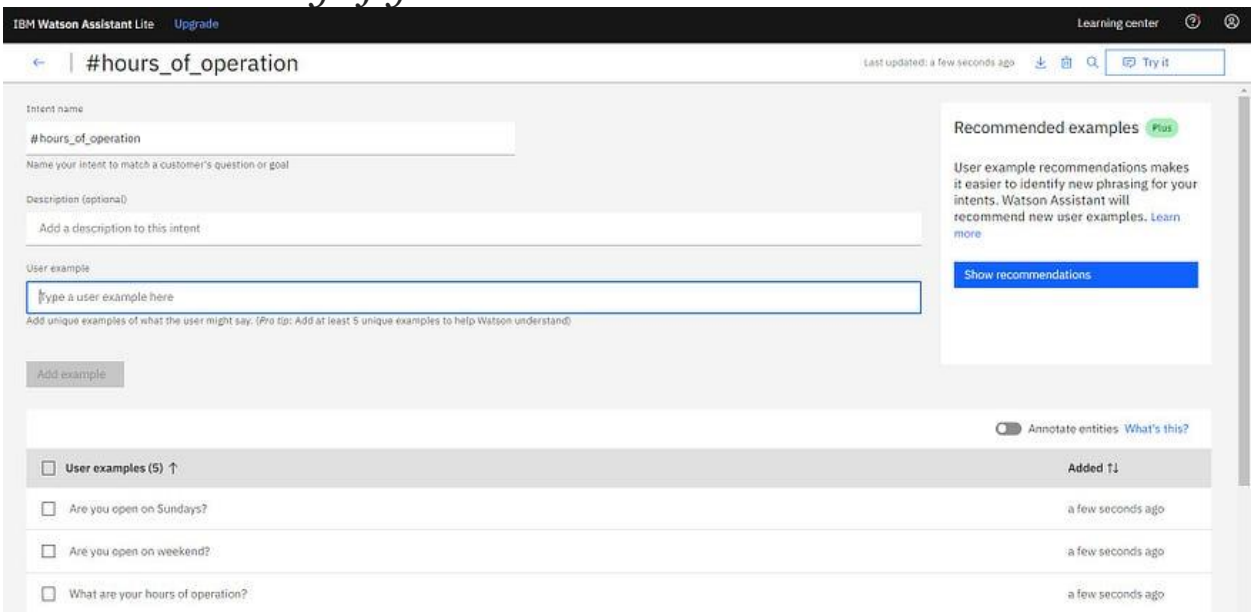
What are your hours of operation?

Are you open on weekend?

Are you open on Sundays?

What are your working hours?

What is the timing of your restaurant?



intent #hours_of_operation created

Till now, we have created an instance of Watson assistant service, added a dialog skill to the assistant, and created two intents. Let's move to the next step.

Step 4: Create Entity and Add to the Skill

In this step, we will create an *Entity* and add it to the skill. But before that, let's understand what an *Entity* is and how it can be used to build a chatbot?

According to Watson Assistant…

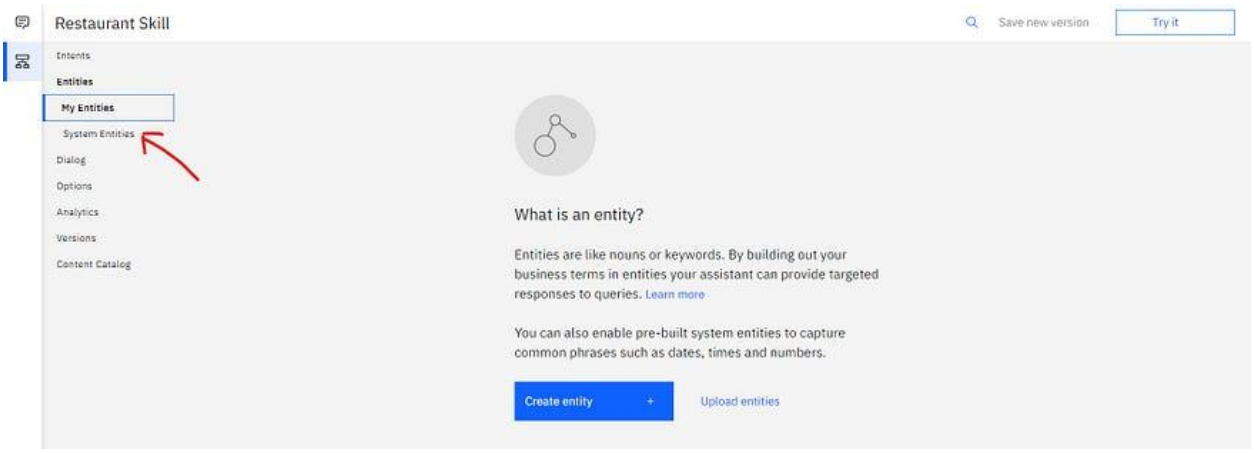
Entities *represent information in the user input that is relevant to the user's purpose*.

If *intents* represent verbs (the action a user wants to do), *entities* represent nouns (the object of, or the context for, that action). For example, when the *intent* is to get a weather forecast, the relevant location and date *entities* are required before the application can return an accurate forecast.

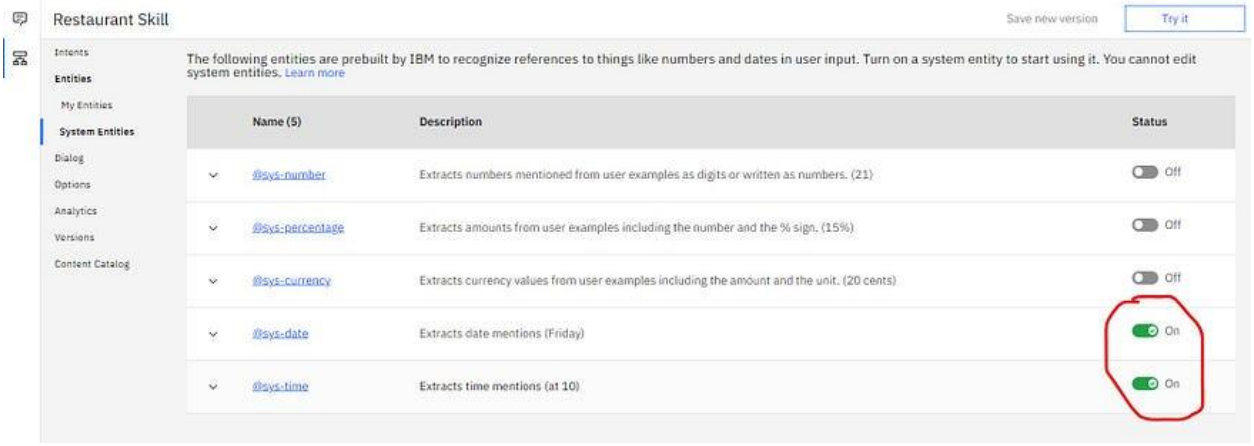
In our example, the user wants to book a table in the restaurant. But to book a table, the chatbot will need to know at what **time** and **date** the user want to book a table. Here **date** and **time** are the *entities* that are required to make a booking of the table.

So we will add two entities, namely, **date** and **time** to the dialog skill. Luckily, we don't need to create these two entities. In Watson Assistant, these two entities are predefined in *System Entities*. We just need to enable these two entities to use.

To enable these two *entities,* click on the "Entities" in the "Restuarant skill". After clicking, You will get the following image.



In the Entities section, you will see two options namely **My Entities** and **System Entities.** Just click on the **System Entities** and enable the *sys-date* and *sys-time* entities to use it in the chatbot.



Enabling sys-date and sys-time entities.

Now you have enabled the **sys-date** and **sys-time** entities. Now you can use them in building our chatbot.

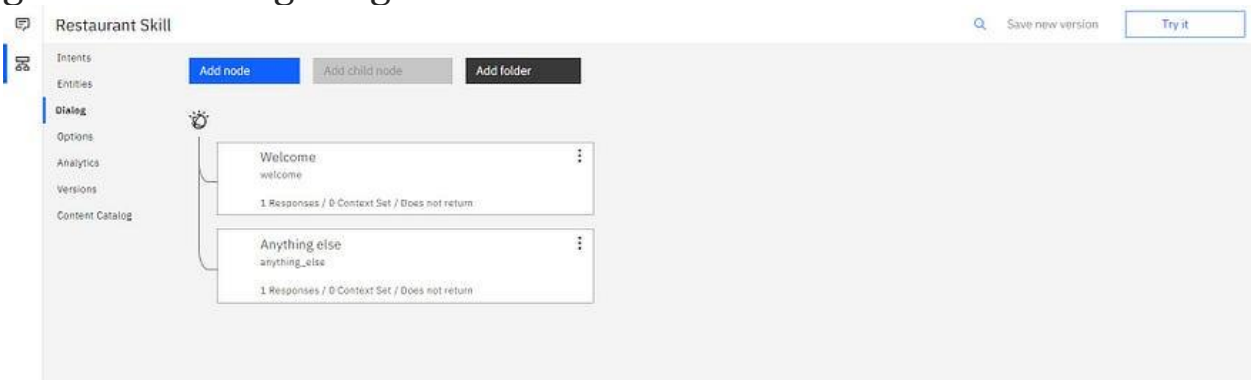
Step 5: Create the Dialog

Now we will build the dialog to give the responses to user input. But before that, let's understand what a **Dialog** is?

According to Watson Assistant…

*A **Dialog** defines the flow of your conversation in the form of a logic tree. It matches intents (what users say) to responses (what your chatbot says back). Each node of the tree has a condition that triggers it, based on user input.*

To build a dialog, click on the “dialog” in the skill menu and you will get the following image.



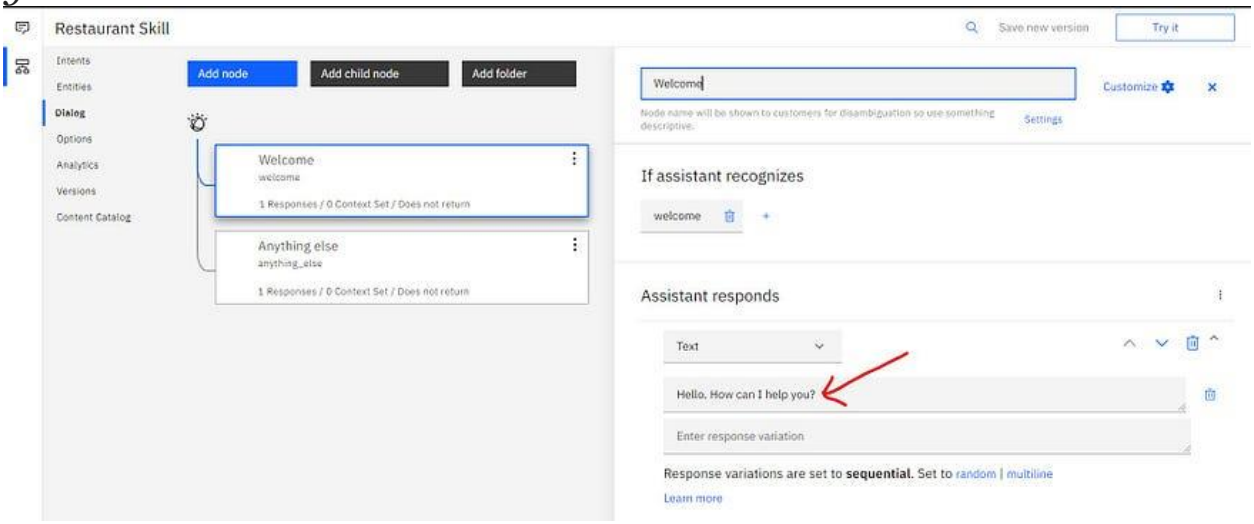
Dialog having Welcome and Anything else node

Here you can see two dialog nodes that are created automatically.

1. **Welcome:** This contains a greeting that is displayed to your users when they first engage with the assistant.
2. **Anything else**: Contains phrases that are used to reply to users when their input is not recognized.

Now click on the **Welcome** node and replace the text with the following written text in the Assistant Responds section.

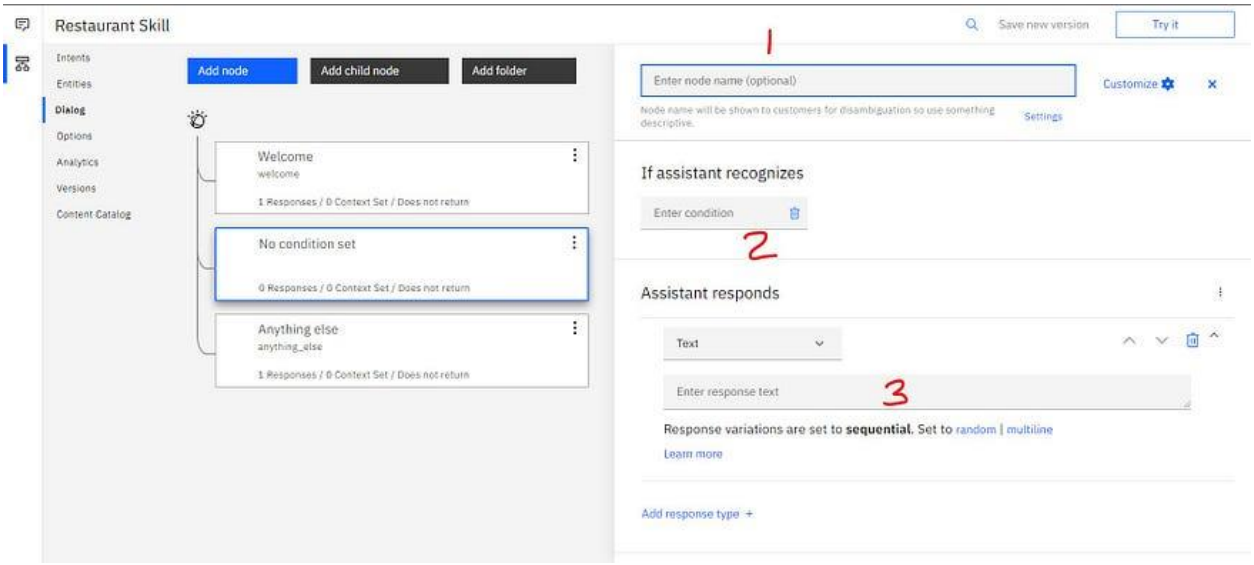
Hello! I am the virtual assistant of the restaurant. How can I help you?



Replace the text

Whenever the user engages with the chatbot, first they will get the above-written message. Now we will need two more dialog nodes, one to give information about hours of operation and one to make a booking.

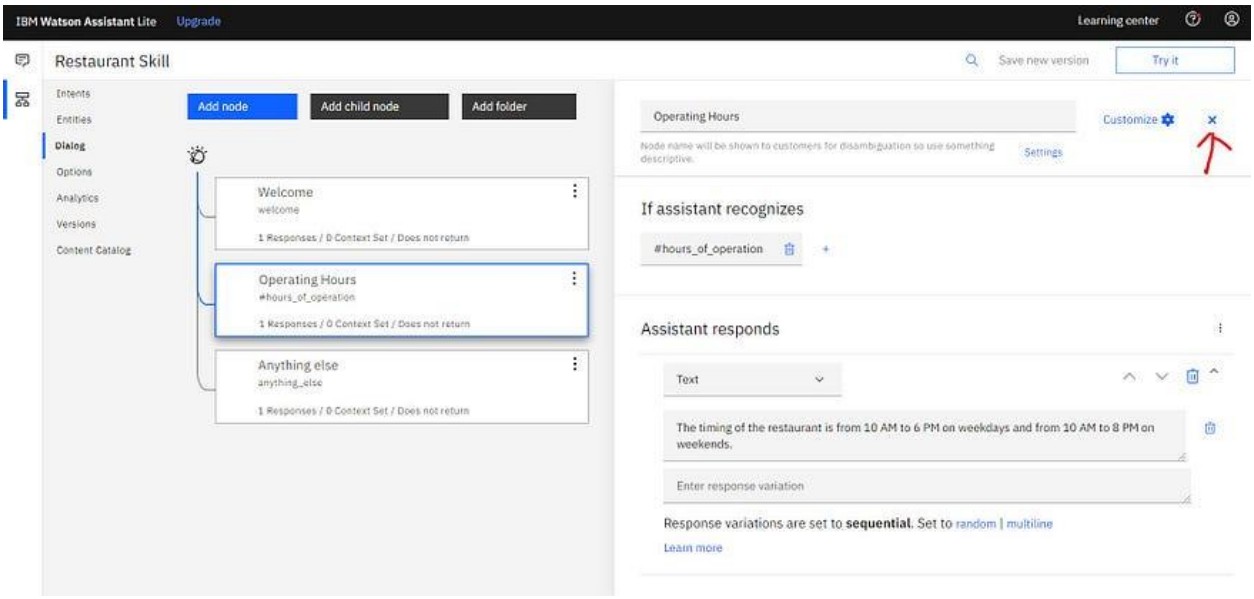
We start with *hours of operation*. To add a new node, click on the three-dot in the upper right corner of the **Welcome** node and click on the *Add node below*. After clicking it you will get the following image.



Now fill in the following detail in the newly created node.

1. **Enter node name(optional)**: Operating Hours
2. **Enter condition**: start typing *#hours_of_operation* and then click on *#hours_of_operation*.
3. **Enter response text**: *The timing of the restaurant is from 10 AM to 6 PM on weekdays and from 10 AM to 8 PM on weekends.*

Now click on the *cross* icon next to ***Customize*** button.



Operating Hours node added to the skill

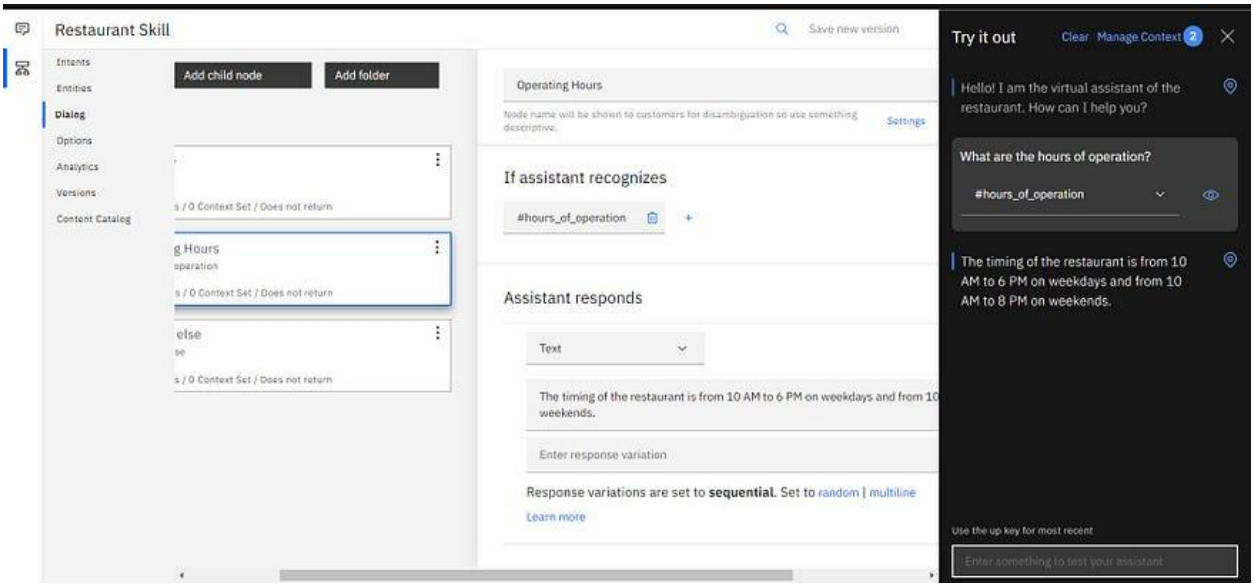
Now whenever a user asks about the working hours of the restaurant, the **Operating Hours** node will be triggered and the user gets the following response:

The timing of the restaurant is from 10 AM to 6 PM on weekdays and from 10 AM to 8 PM on weekends.

Now you can try the chatbot that we have built now. Just click on **Try it** in the upper right corner and type the following sentence.

What are the hours of operation?

You will get the following response.

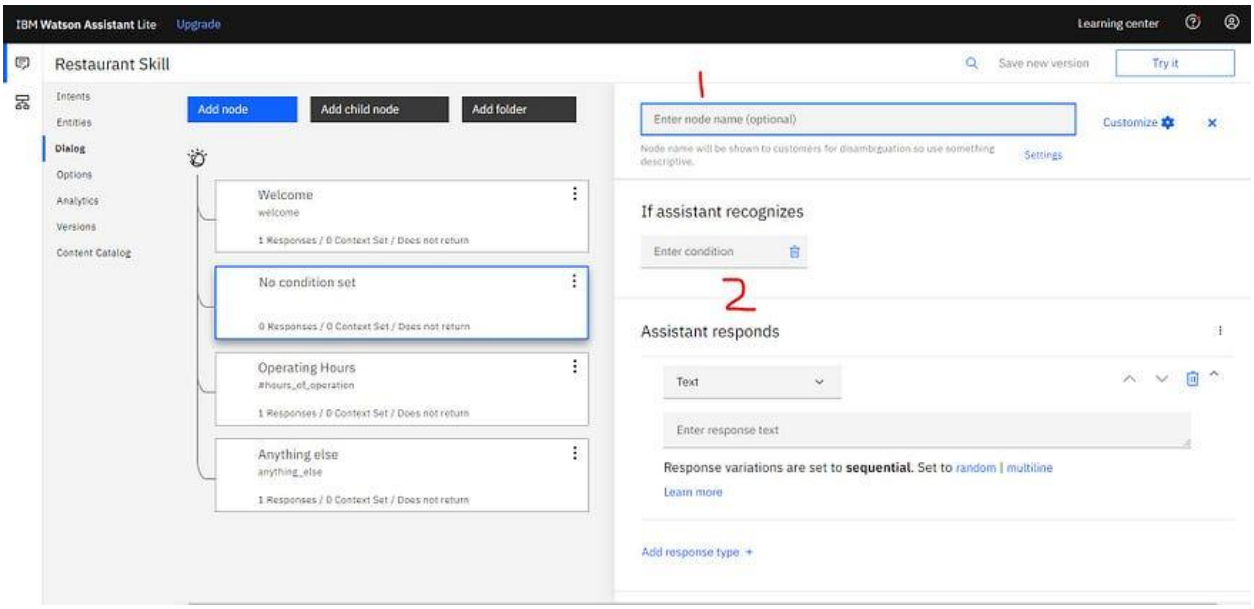


Trying the Chatbot

Till now, we have added the dialog node to give the information about the hours of operation. Now we will add our last dialog node to **make a booking** for the table in the restaurant.

To add a new dialog node, just click on the three-dots in the **Operating Hours** node and click on the *Add node above*.

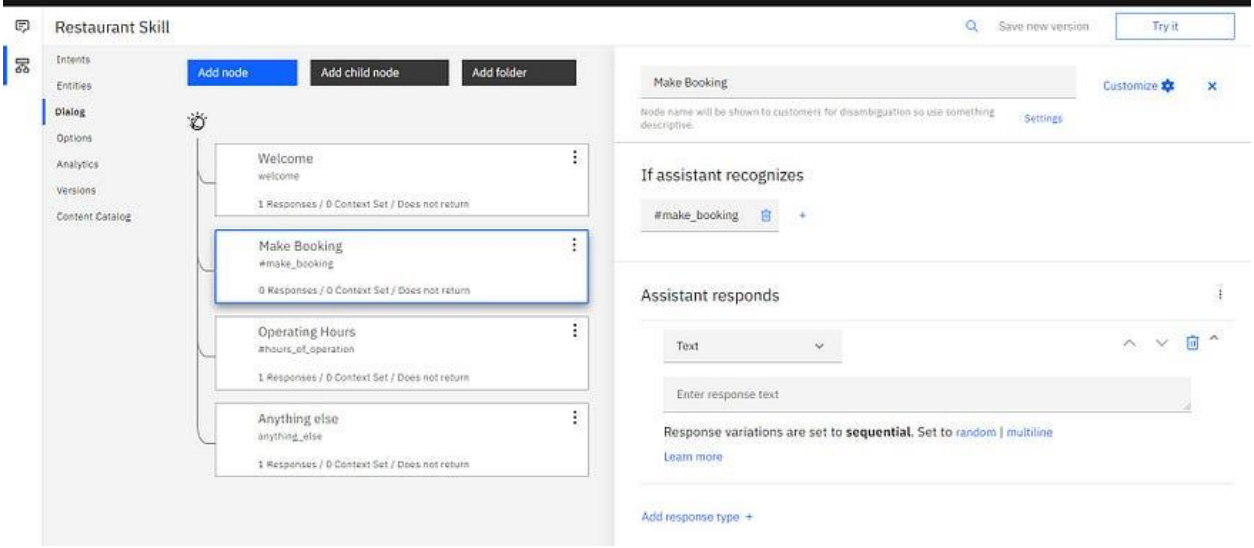
You will get the following image.



Dialog node for make a booking

Now fill in the following details in the newly created dialog node.

1. **Enter node name(optional)**: Make Booking
2. **Enter condition**:start typing *#make_booking* and then click on *#make_booking*.



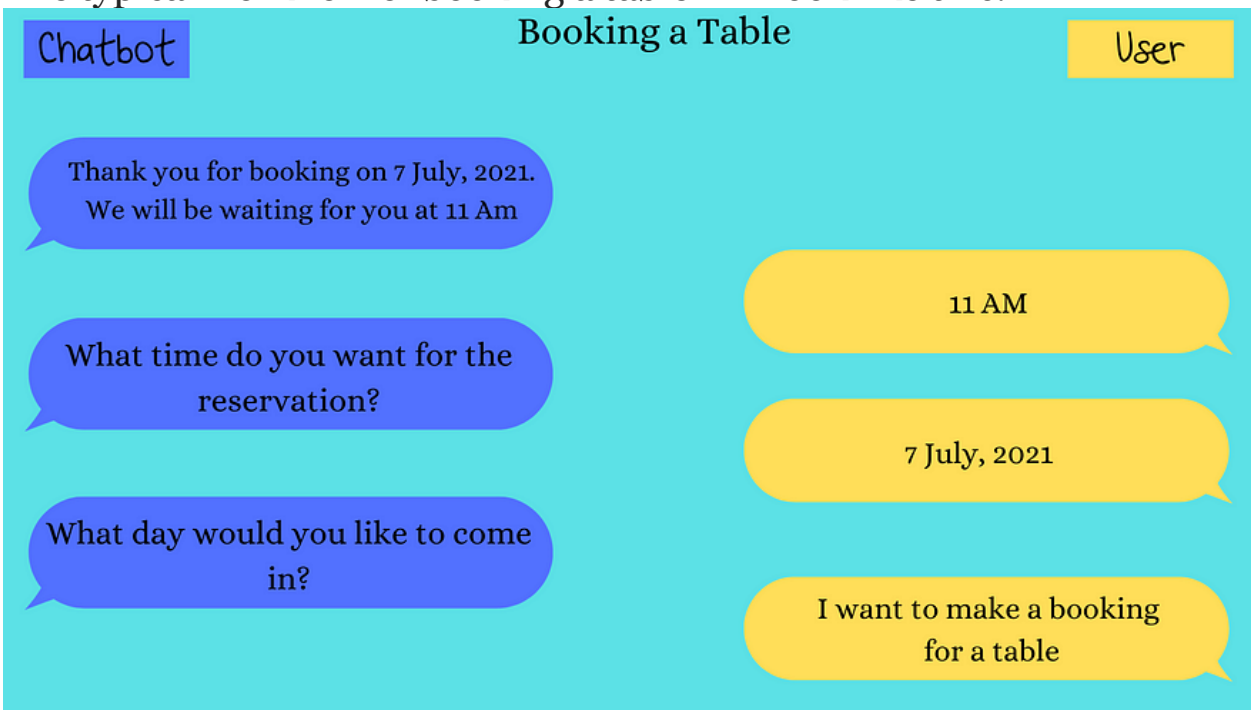
Node after making change.

We have entered the ***#make_booking*** intent as a condition to recognize so whenever a user asks about booking a table, the chatbot will recognize the intent ***#make_booking*** and this node will be triggered.

To book a table in the restaurant, the chatbot needs to ask the user two things.

1. **What day would you like to come in?** (date of the booking)
2. **What time do you want for the reservation?** (time of the booking)

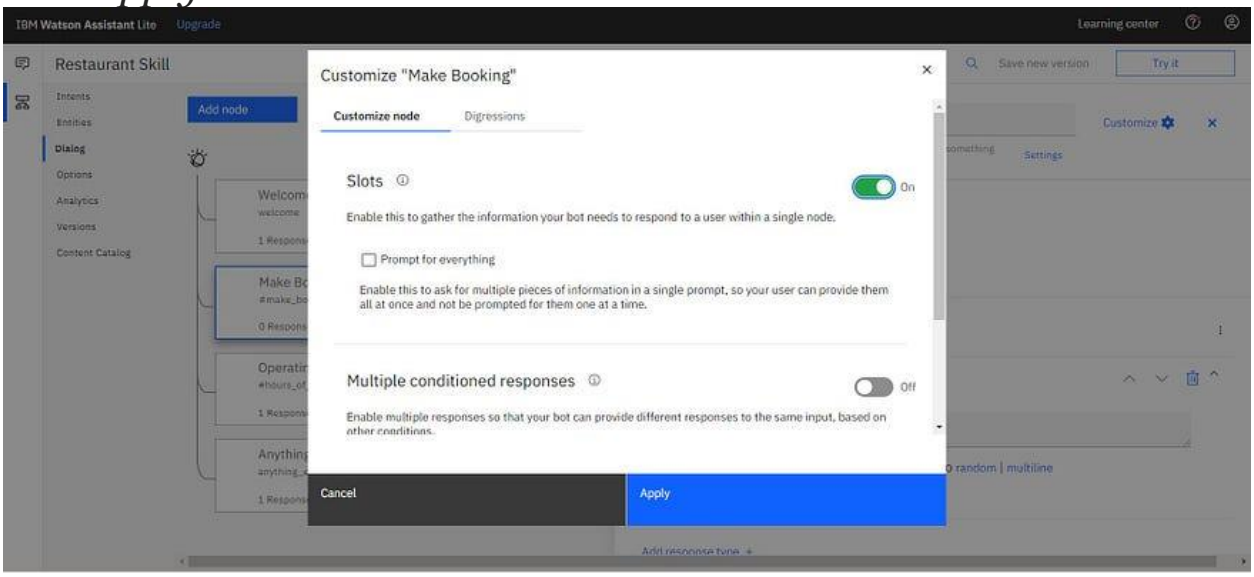
The typical workflow of booking a table will look like this.



Workflow of Booking a table through the chatbot

To create this workflow, we will use the **Slots**. Slots provide a structured format through which you can ask for and save multiple pieces of information from a user within a single node. They are most useful when you have a specific task in mind and need key pieces of information from the user before you can perform it.

To use the **slots,** click on ***Customize*** button in the upper right corner of ***Make Booking*** *node*, set the Slots switch to On, and then click *Apply*.



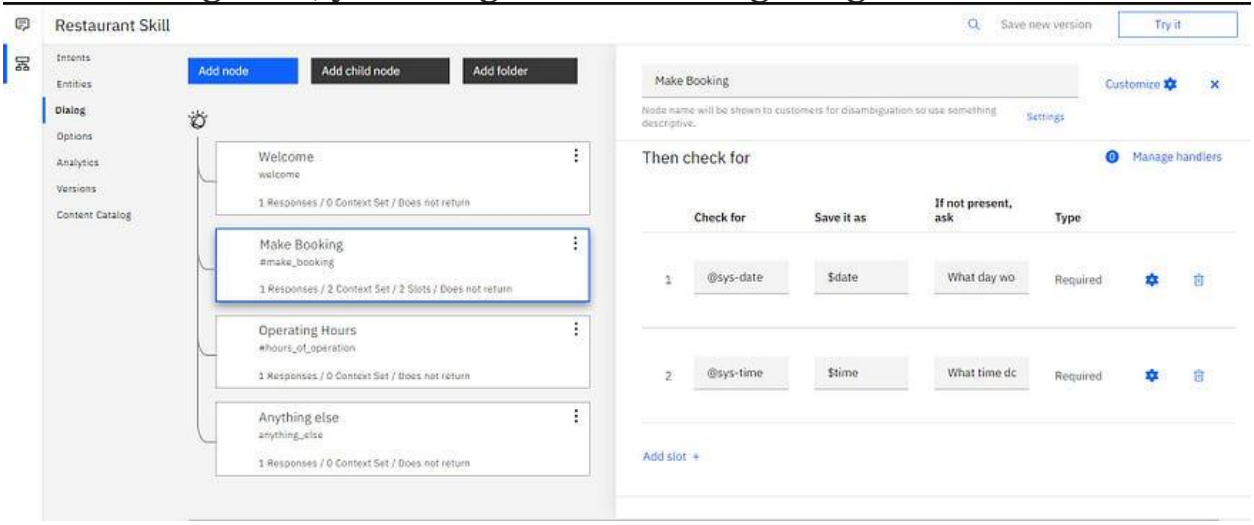
set the Slots switch to On and click Apply

In ***Then check for*** *Section,* Add the following slots :

1. **check for** — @sys-date, **save as** — $date, **If not present, ask** — What day would you like to come in?

2. **check for** — @sys-time, **save as** — $time, **If not present, ask** — What time do you want for the reservation?

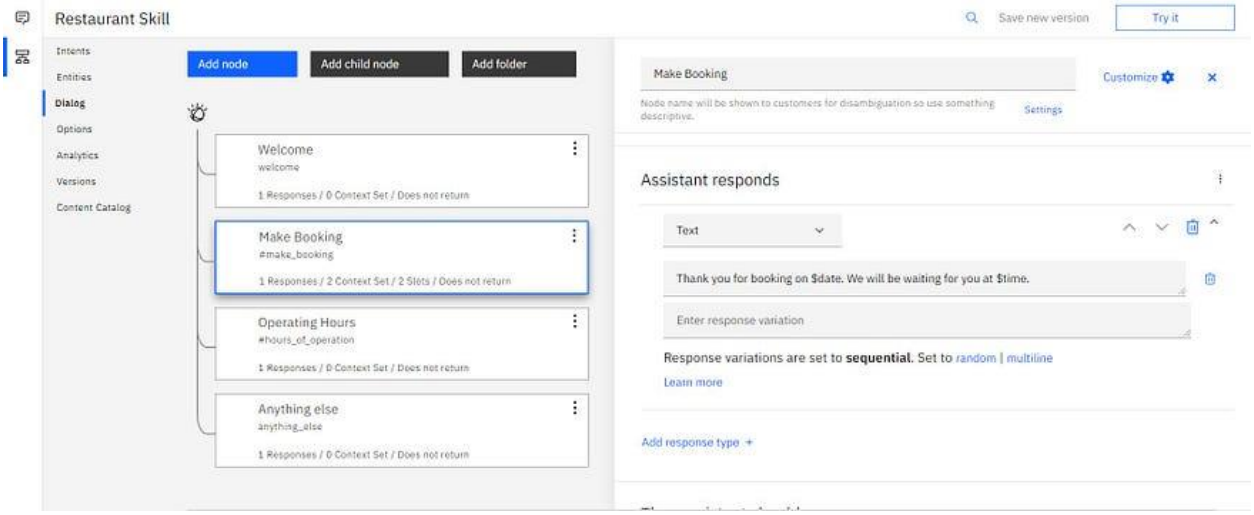
After adding slots, you will get the following image.



Adding slots to the node.

Now, In the **Assistant responds** section, enter the following text.

Thank you for booking on $date. We will be waiting for you at $time.



Adding text response to the node.

The chatbot is complete now. You can try it using the **Try it** option in the upper right corner.

Sample source code

```
import random
import json
import pickle
import numpy as np

import nltk
from nltk.stem import WordNetLemmatizer

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import SGD

lemmatizer = WordNetLemmatizer()

intents = json.loads(open('intents.json').read())

words = []
classes = []
documents = []
ignore_letters = ['?', '!',',','.']

for intent in intents['intents']:
    for pattern in intent['patterns']:
```

```
        word_list = nltk.word_tokenize(pattern)
        words.extend(word_list)
        documents.append((word_list,intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

words = [lemmatizer.lemmatize(word) for word in words if word not in ignore_letters]
words = sorted(set(words))

classes = sorted(set(classes))

pickle.dump(words, open('words.pkl', 'wb'))
pickle.dump(classes, open('classes.pkl', 'wb'))

training = []
output_empty = [0] * len(classes)

for document in documents:
    bag =[]
    word_patterns = document[0]
    word_patterns = [lemmatizer.lemmatize(word.lower()) for word in word_patterns]
    for word in words:
        bag.append(1) if word in word_patterns else bag.append(0)

    output_row = list(output_empty)
```

```
    output_row[classes.index(document[1])] = 1
    training.append([bag, output_row])

random.shuffle(training)
training = np.array(training)

train_x = list(training[:, 0])
train_y = list(training[:, 1])

model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
model.save('chatbotmodel.h5', hist)

print('Done')
```