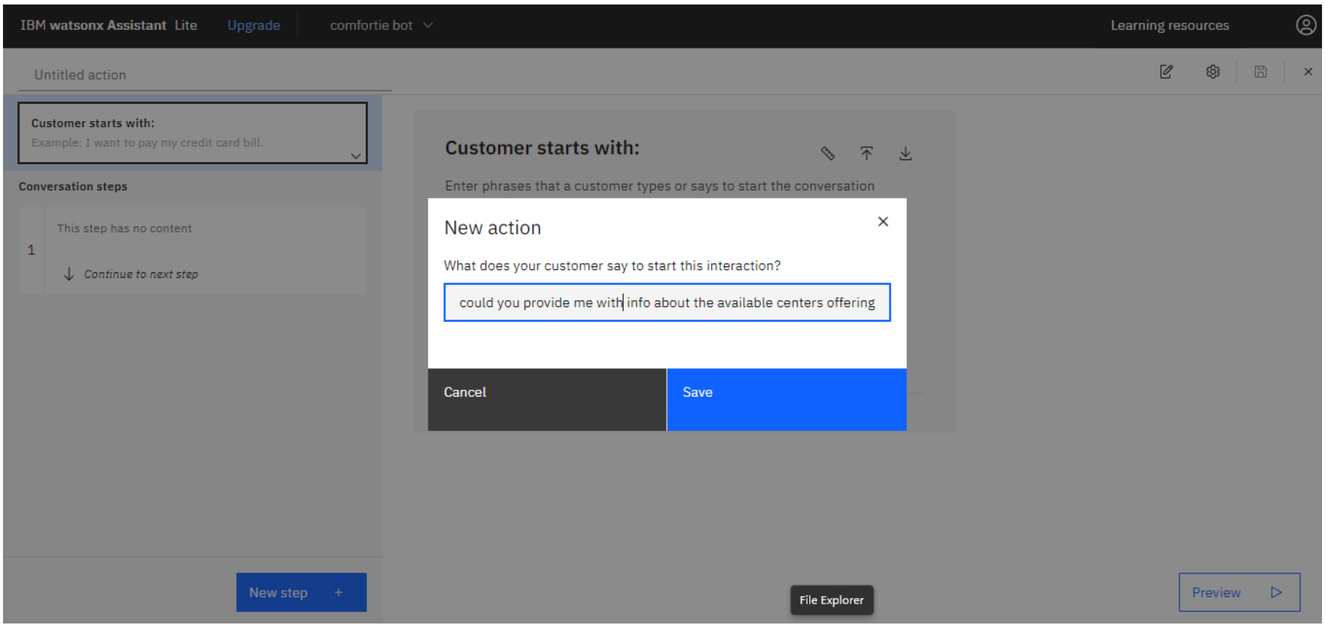
IBM CHATBOT WATSON ASSISSTANT-FINAL SUBMISSION

Outline the project's objective, design thinking process, and development phases. Describe the chatbot's persona, conversation flow, and technical implementation using Watson Assistant. Provide examples of user queries and the chatbot's responses.

The objective of the project is to Adopt IBM Cloud Watson Assistant as Chabot development platform, leveraging its robust natural language processing capabilities and integration features.Clearly define the primary objectives of the Chabot, including handling routine inquiries, reducing response times, and assisting with common issues.Develop intuitive conversational flows for the Chabot to ensure seamless interactions with users.

- The Chatbot would be designed perfectly in the parts of intents,entities,actions and response to achieve the problem design.
- We define actions by specifying pre-defined commands or tasks that users can ask the Chatbot to perform. These actions should be welldefined to ensure the Chatbot provides clear and effective responses.
- We provide new actions to get response flow from Watson assistant.When the user matches their requests with the actions,the assistant provides the specified response.



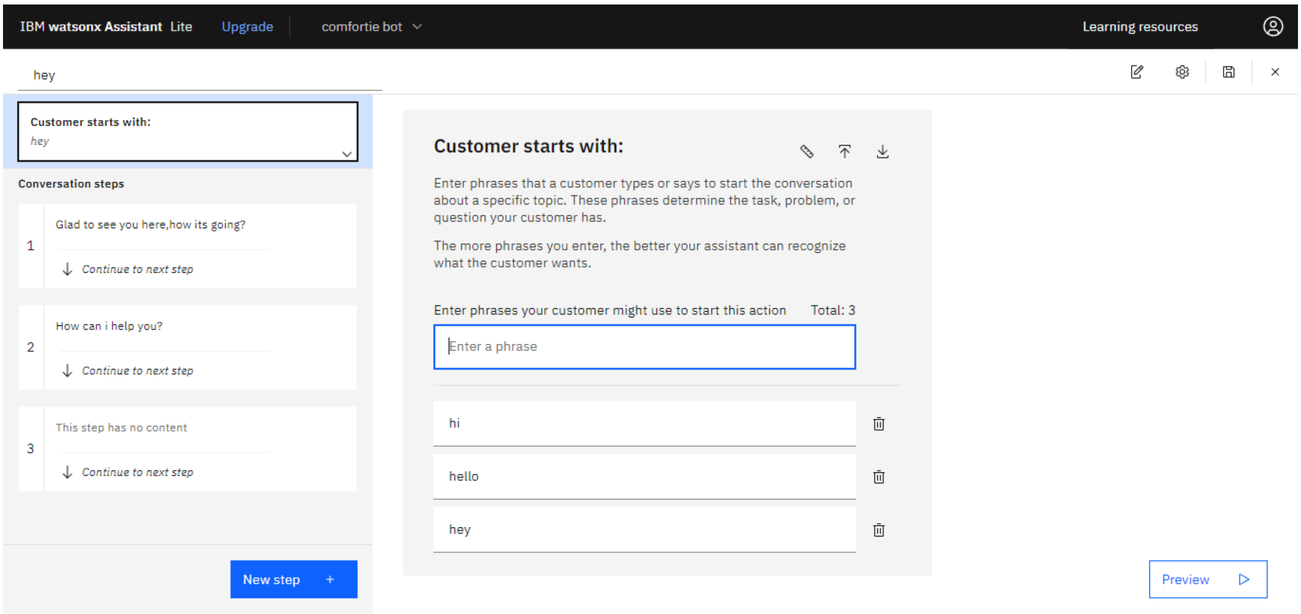
Customer starts with:

Enter phrases that a customer types or says to start the conversation

New action

What does your customer say to start this interaction?

could you provide me with info about the available centers offering



hey

Customer starts with:
hey

Conversation steps

1 Glad to see you here,how its going?
Continue to next step

2 How can i help you?
Continue to next step

3 This step has no content
Continue to next step

Customer starts with:

Enter phrases that a customer types or says to start the conversation about a specific topic. These phrases determine the task, problem, or question your customer has.

The more phrases you enter, the better your assistant can recognize what the customer wants.

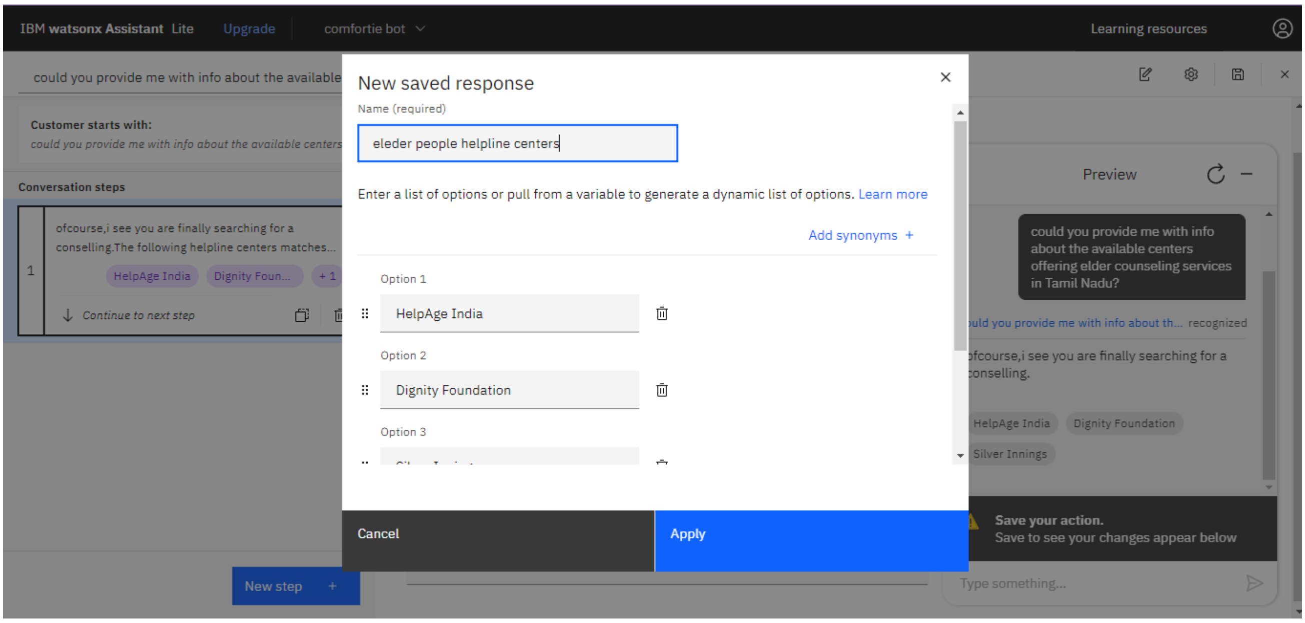
Enter phrases your customer might use to start this action Total: 3

Enter a phrase

hi

hello

hey



could you provide me with info about the available

Customer starts with:
could you provide me with info about the available centers

Conversation steps

1 ofcourse,i see you are finally searching for a conselling.The following helpline centers matches...
HelpAge India Dignity Foun... +1
Continue to next step

New saved response

Name (required)

eleder people helpline centers

Enter a list of options or pull from a variable to generate a dynamic list of options. Learn more

Add synonyms +

Option 1
HelpAge India

Option 2
Dignity Foundation

Option 3

Preview

could you provide me with info about the available centers offering elder counseling services in Tamil Nadu?

uld you provide me with info about th... recognized

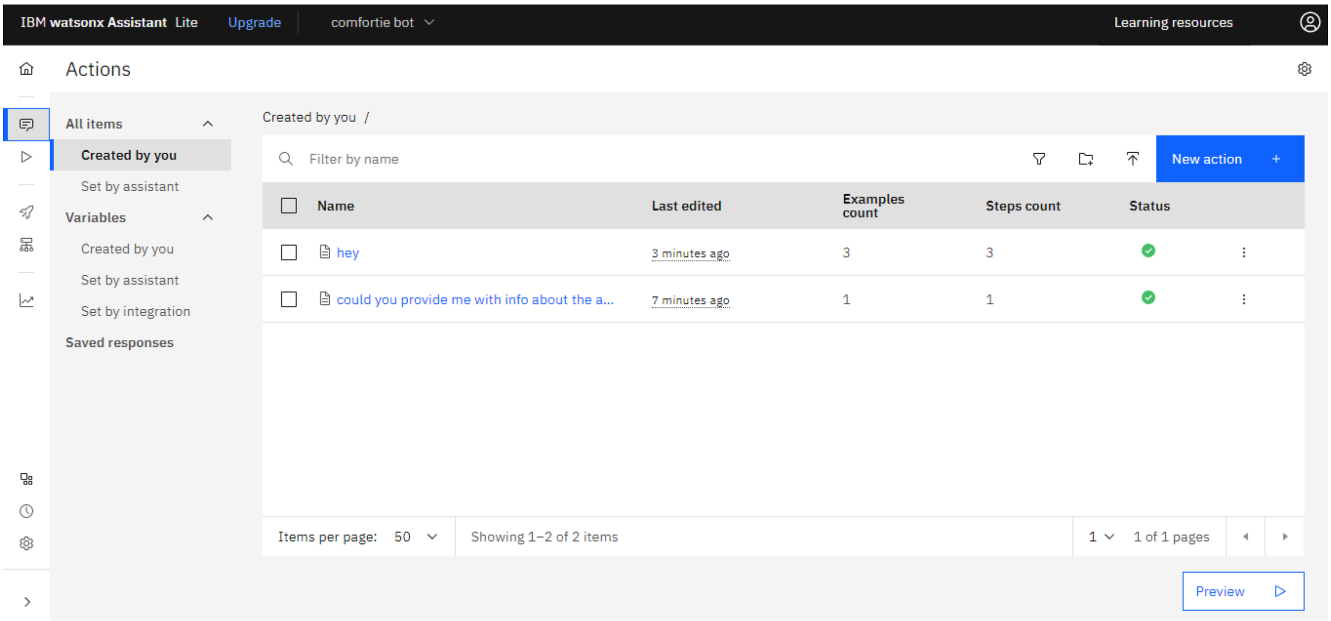
ofcourse,i see you are finally searching for a conselling.

HelpAge India Dignity Foundation
Silver Innings

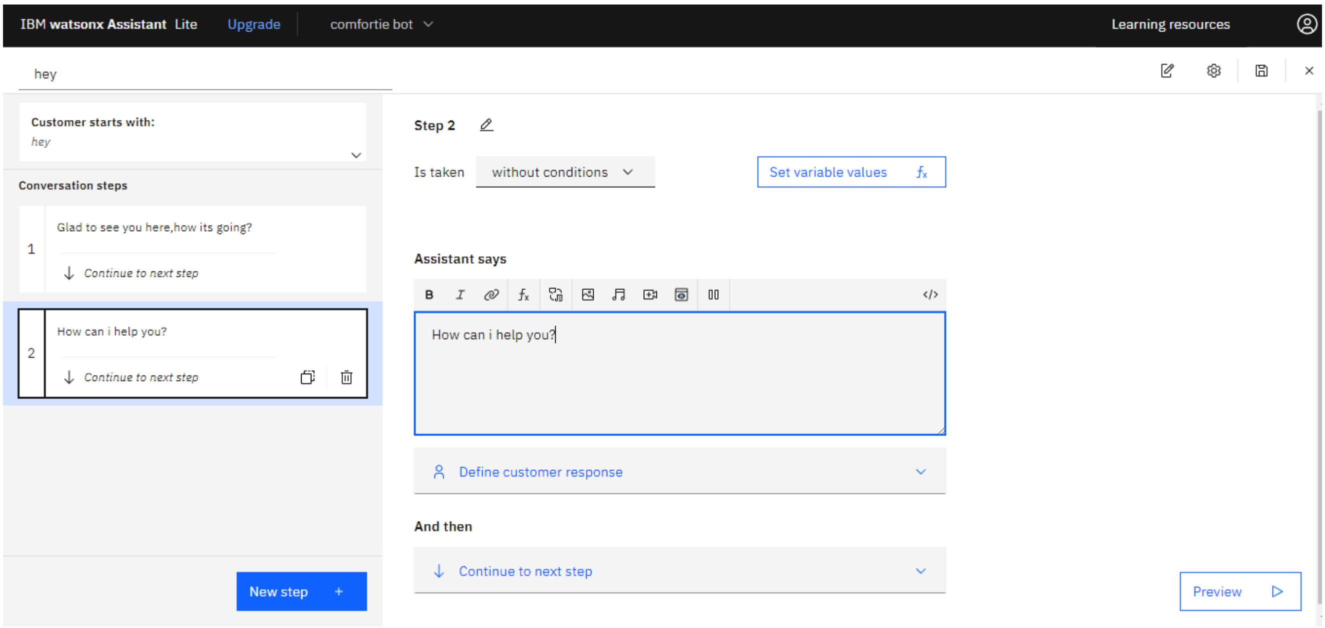
Save your action.
Save to see your changes appear below

Type something...

- We can provide multiple options to be a response from the Chatbot assisstent.
- When defining actions in Watson Assistant, you have the option to specify a range of responses and follow-up actions that the Chatbot can take based on user input, enabling dynamic and context-aware interactions.



☐	Name	Last edited	Examples count	Steps count	Status
☐	hey	3 minutes ago	3	3	✓
☐	could you provide me with info about the a...	7 minutes ago	1	1	✓



Customer starts with:
hey

Conversation steps

1 Glad to see you here,how its going?
Continue to next step

2 How can i help you?
Continue to next step

Step 2

Is taken without conditions

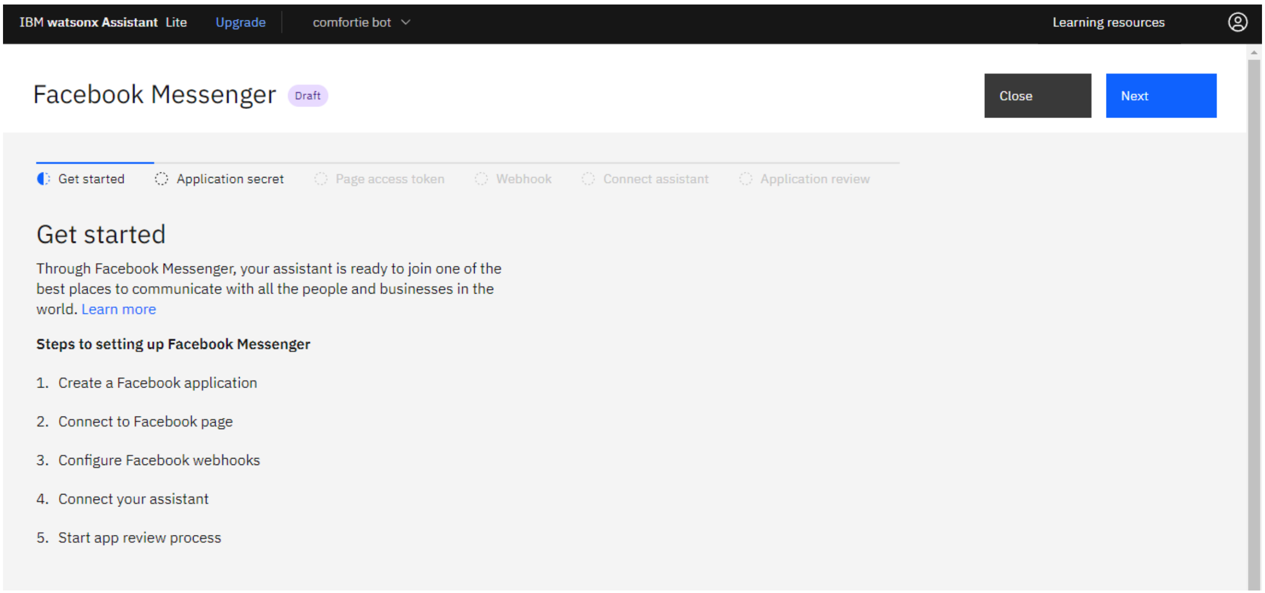
Assistant says

How can i help you?

And then

Continue to next step

- To integrate your chatbot with Facebook Messenger, you can use the Facebook Messenger platform as a channel to deploy your Watson Assistant chatbot.In the IBM Watson Assistant, navigate to the "Integrate" section, and choose Facebook Messenger as one of the channels you want to integrate with.



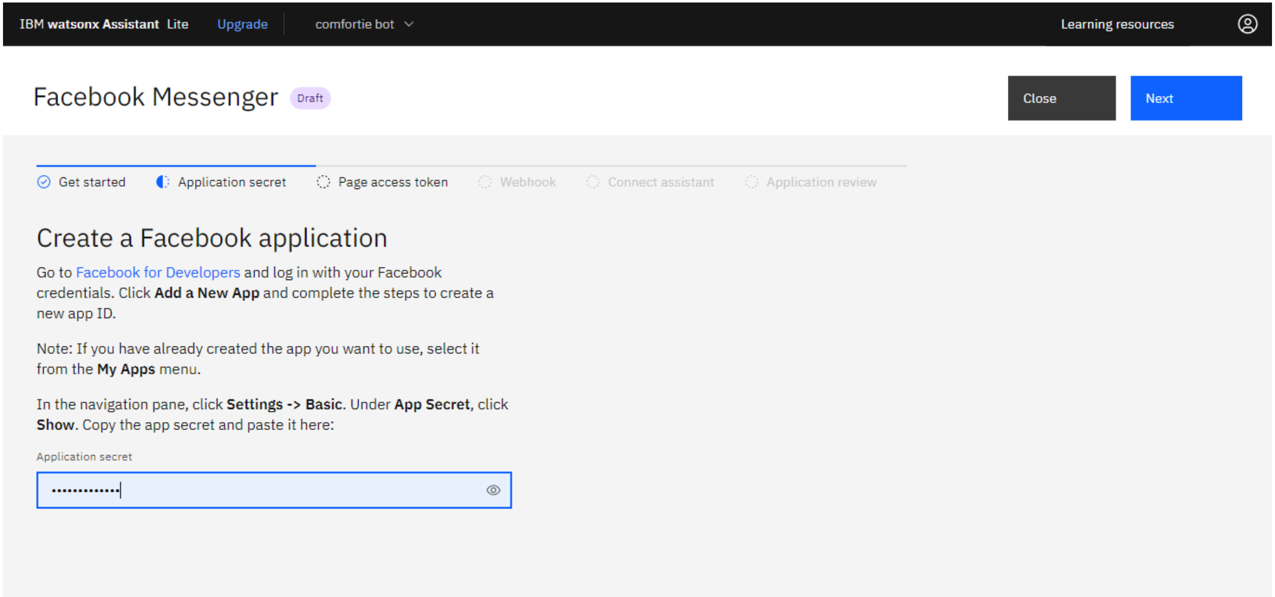
Facebook Messenger Draft

Get started

Through Facebook Messenger, your assistant is ready to join one of the best places to communicate with all the people and businesses in the world. Learn more

Steps to setting up Facebook Messenger

1. Create a Facebook application
2. Connect to Facebook page
3. Configure Facebook webhooks
4. Connect your assistant
5. Start app review process



Facebook Messenger Draft

Create a Facebook application

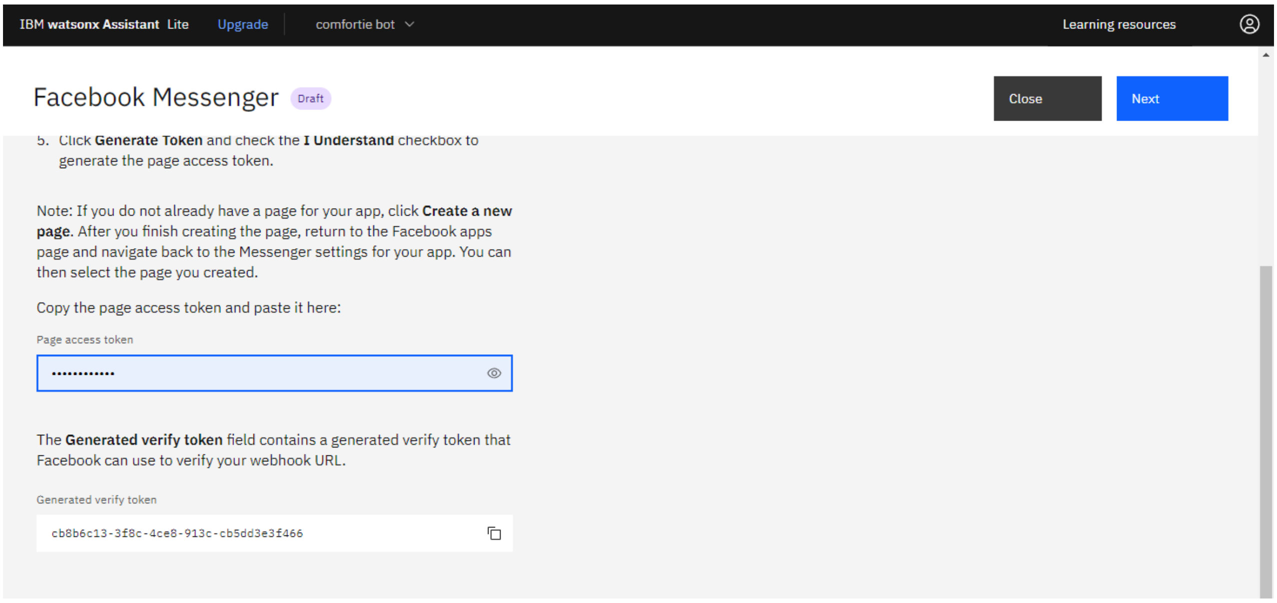
Go to Facebook for Developers and log in with your Facebook credentials. Click **Add a New App** and complete the steps to create a new app ID.

Note: If you have already created the app you want to use, select it from the **My Apps** menu.

In the navigation pane, click **Settings -> Basic**. Under **App Secret**, click **Show**. Copy the app secret and paste it here:

Application secret

•••••••••••••



Facebook Messenger Draft

5. Click **Generate Token** and check the **I Understand** checkbox to generate the page access token.

Note: If you do not already have a page for your app, click **Create a new page**. After you finish creating the page, return to the Facebook apps page and navigate back to the Messenger settings for your app. You can then select the page you created.

Copy the page access token and paste it here:

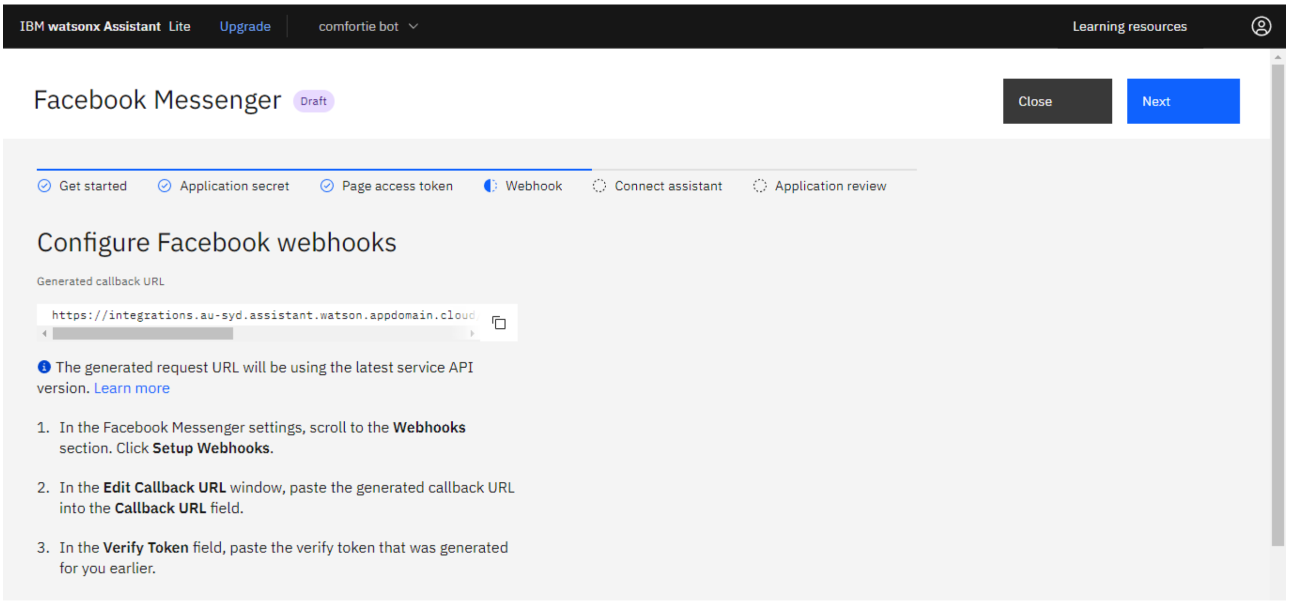
Page access token

••••••••••••

The **Generated verify token** field contains a generated verify token that Facebook can use to verify your webhook URL.

Generated verify token

cb8b6c13-3f8c-4ce8-913c-cb5dd3e3f466



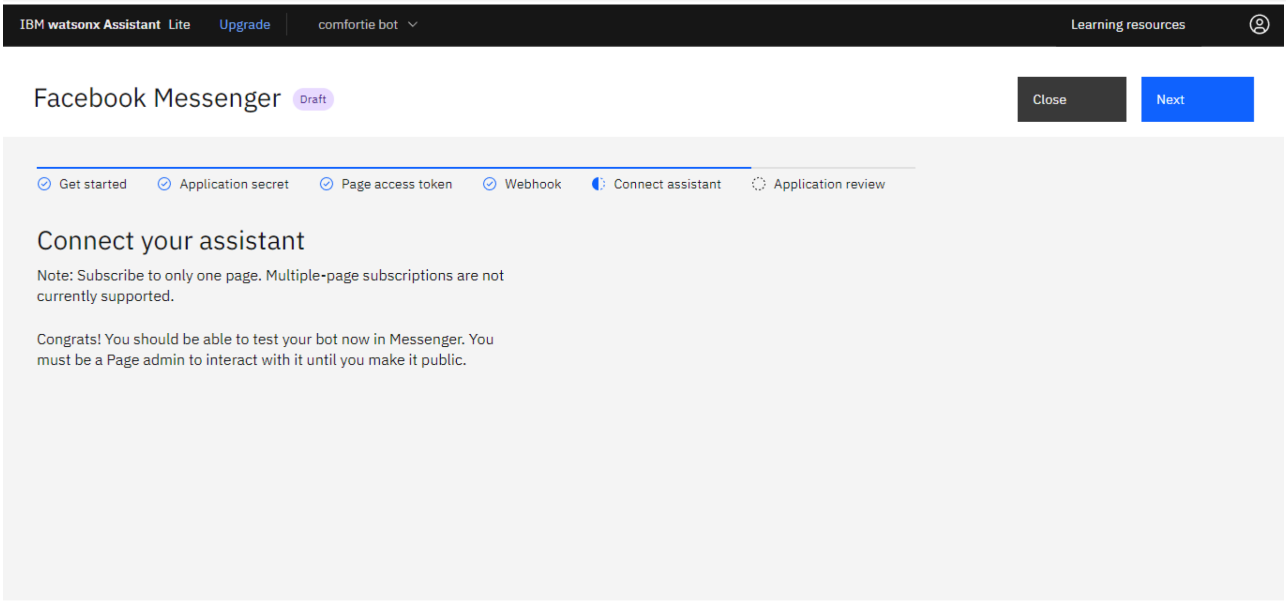
Configure Facebook webhooks

Generated callback URL

```
https://integrations.au-syd.assistant.watson.appdomain.cloud
```

The generated request URL will be using the latest service API version. Learn more

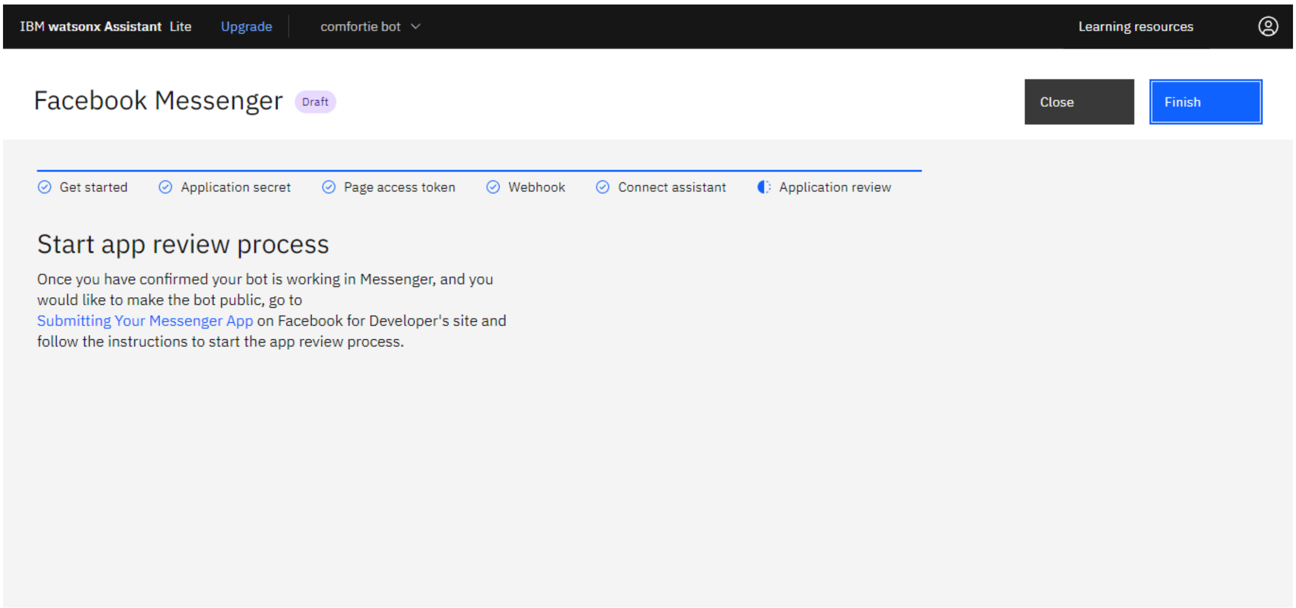
1. In the Facebook Messenger settings, scroll to the **Webhooks** section. Click **Setup Webhooks**.
2. In the **Edit Callback URL** window, paste the generated callback URL into the **Callback URL** field.
3. In the **Verify Token** field, paste the verify token that was generated for you earlier.



Connect your assistant

Note: Subscribe to only one page. Multiple-page subscriptions are not currently supported.

Congrats! You should be able to test your bot now in Messenger. You must be a Page admin to interact with it until you make it public.

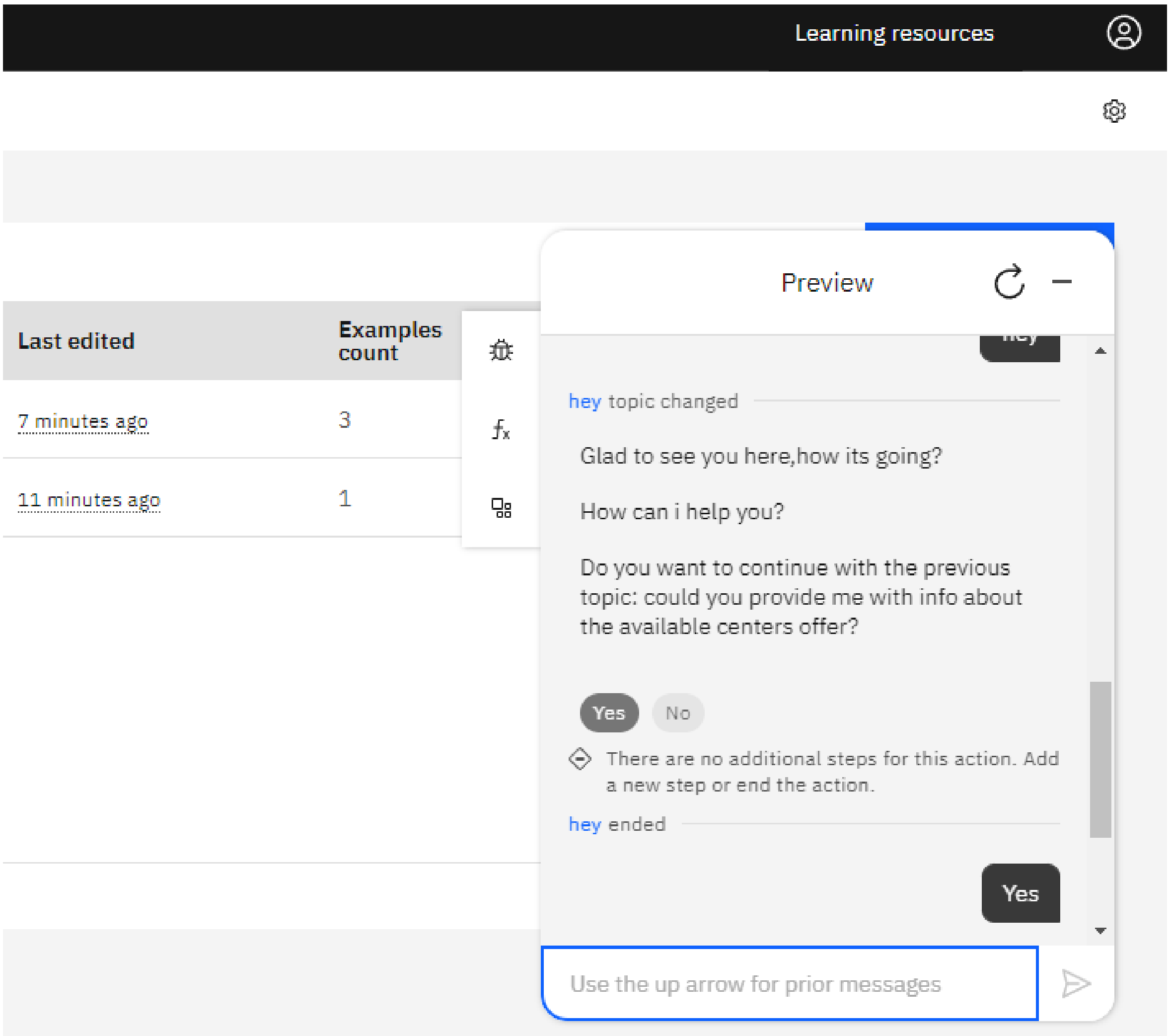


Start app review process

Once you have confirmed your bot is working in Messenger, and you would like to make the bot public, go to Submitting Your Messenger App on Facebook for Developer's site and follow the instructions to start the app review process.

- To integrate your chatbot with Slack, you can use the Slack platform as a channel to deploy your Watson Assistant chatbot.

- IBM Watson Assistant comes with built-in NLP capabilities, but you can enhance them by training your assistant to better understand user intents and entities. You can also integrate additional NLP services if needed.



Last edited	Examples count
7 minutes ago	3
11 minutes ago	1

hey topic changed

Glad to see you here,how its going?

How can i help you?

Do you want to continue with the previous topic: could you provide me with info about the available centers offer?

There are no additional steps for this action. Add a new step or end the action.

hey ended

Yes

- Test your chatbot on both the Facebook Messenger and Slack platforms to ensure it functions correctly.
- Continually refine and update the chatbot's dialog based on user interactions and feedback. Monitor your chatbot's performance using analytics and user feedback.

Code is given below:

```
import random

import json

import pickle

import numpy

import tensorflow

import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

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

```
words = []
classes = []
documents = []
ignoreLetters = ['?','!','.',',']
for intent in intents['intents']:
for pattern in intent['patterns']:
wordList = nltk.word_tokenize(pattern)
words.extend(wordList)
documents.append((wordList, intent['tag']))
if intent['tag'] not in classes:
classes.append(intent['tag'])
words = [lemmatizer.lemmatize(word) for word in  words if word not in ignoreLetters]
words = sorted(set(classes))
classes= sorted(set(classes))
pickle.dump(words, open('words.pkl', 'wb'))
pickle.dump(classes, open('classes.pkl', 'wb'))
training = []
outputEmpty = [0] * len(classes)
for document in documents:
bag= []
wordPatterns = document[0]
wordPatterns = [lemmatizer.lemmatize(word.lower()) for word in wordPatterns]
for word in words: bag.append(1) if word in wordPatterns else bag.append(0)
outputRow = list(outputEmpty)
outputRow[classes.indexdocument[1]] = 1
training.append(bag + outputRow)
random.shuffle(training)
training = np.array(training)
trainx = training[:, :len(words)]
trainY = training[:, :len(words):]
model = tf.keras.Sequential()
model.add(tf.keras.layer.Dense(128, input_shape = (len(trainx[0]),),activation = 'relu' ))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(64, activation = 'relu'))
model.add(tf.keras.layers.Dense(len(trainY, activation= "softmax")))
```

```
sgd = tf.keras.optimizers.SGD(learning_rate=0.01, momentum = 0.9, nseterov=True)
model.compile(loss= 'categorical-croossentropy', optimizer = sgd, metrices= ['accuracy'])
hist = model.fit(np.array(trainX), np.array(trainY, epochs = 200, batch_size = 5, verbose=1))
model.save('chatbot_simplilearnmodel.h5', hist)
print("Executed")
```

P**reprocessing a dataset for training a chatbot in Python typically involves several key steps**:

- Text Tokenization: Splitting the text into individual words or tokens. You can use libraries like NLTK for this.

- Text Lowercasing: Converting all text to lowercase to ensure consistency in text representation.

- Stop Word Removal: Eliminating common words (e.g., "the," "and") that don't carry much meaning and can be ignored. NLTK have built-in lists of stop words.

- Special Character Removal: Removing punctuation and other special characters that may not be relevant for chatbot training.

- Lemmatization or Stemming: Reducing words to their base form to handle variations (e.g., "running" to "run"). NLTK provide lemmatization and stemming options.

- Handling Contractions: Expanding contractions like "can't" to "cannot" for consistent representation.s