Build a chatbot project by using Watson Assistant

Chatbot projects that use Watson Assistant involve three phases: scope, design, and integrate.

In the scope phase, you gather requirements for the conversation and how customers support the use case today. They might have a script, coded procedures, or other artifacts.

You define personas, create an empathy map, and build a system context diagram. Then, you extract the potential list of intents. Intents are the purposes or goals that are expressed in a user's input, such as answering a question or processing a payment. After you define intents, you assess the sentences that lead to those intents.

In the design phase, you create an instance of Watson Assistant and use its builder tool to define the intents and the entities. An entity represents a class of object or data type that is relevant to a user's purpose. At the end of the design phase, you start the dialog flow and unit-test it.

Finally, in the integrate phase, you develop the web app or microservice that interacts with Watson Assistant. You implement the business logic to handle the conversation context, and add other components to complement the business requirements, such as the IBM Watson Retrieve and Rank Service, ODM business rules, or IBM BPM process.

Tasks

When you implement a cognitive solution, be sure to apply Enterprise Design Thinking to develop an innovative business-impact application.

Tutorial

An IT support director wants to modernize the way that the team supports internal staff and relieve the team of routine tasks. Currently, 20,000 tickets are issued each year. Half of the calls are answered in 7 minutes; however, many situations take up to 70 minutes to resolve. Those situations result in frustration and a loss of productivity for the support staff.

In this tutorial, you implement a simple IT support help-me conversation chatbot. The chatbot streamlines IT support queries by automating the dialog flow.

You complete these tasks:

Gain a basic understanding of Watson Assistant.
Understand the development steps.
Apply design thinking for a cognitive solution.
Create an instance of Watson Assistant and a workspace.
Define intent and entities to help natural language processing.
Build a simple dialog flow.
Use the context object for more advanced dialog.
Develop a hierarchical flow.
Use variables to get data from Watson Assistant.
Use the API.
This tutorial is structured in layers. If you are a beginner, focus on developing Watson Assistant artifacts and test within the Watson Assistant service. If you are more advanced, you can also study and tune the broker code to support more advanced features.

Prerequisites

You must have an IBM Cloud account. The account is free and provides access to everything you need to develop, track, plan, and deploy apps. Sign up for a trial. The account requires an IBMid. If you don't have an IBMid, you can create one when you register.

Your IBM Cloud account must have at least 8 GB of runtime and container memory, plus access to provision up to 10 services.

You need a clone of this GitHub repository, as you might need some of the files to expedite your work.

To develop on the broker code or to run locally or on IBM Cloud, you must be familiar with Node.js, Express.js, and Angular 2.

Task 1: Create the Assistant service

Task 2: Create a workspace

Task 3: Create intents

Task 4: Test the intents

Task 5: Add entities

Task 6: Build the dialog

Task 7: Complete advanced dialog work

Task 8: Use the API

```
import json
from ibm_watson import AssistantV2
from ibm_cloud_sdk_core.authenticators import
IAMAuthenticator
```

INPUT CODE

```
# Replace with your Watson Assistant credentials
api_key = "YOUR_API_KEY"
assistant_id = "YOUR_ASSISTANT_ID"
url = "YOUR_ASSISTANT_URL"

# Set up the Watson Assistant client
authenticator = IAMAuthenticator(api_key)
assistant = AssistantV2(
    version='2021-06-14',
    authenticator=authenticator
)
assistant.set_service_url(url)

# Define a function to send a message to Watson
```

```
Assistant
def send_message(message_input):
    response = assistant.message(
        assistant_id=assistant_id,
        input={
            'message_type': 'text',
            'text': message_input
        }
    ).get_result()
    return response

# Main loop for interacting with the chatbot
print("Chatbot: Hello! How can I assist you today?")
while True:
    user_input = input("You: ")

    if user_input.lower() == 'exit':
        print("Chatbot: Goodbye!")
        break

    response = send_message(user_input)
    chatbot_response = response['output']['generic'][0]['text']
    print(f"Chatbot: {chatbot_response}")
```

OUTPUT CODE

```
import json
from ibm_watson import AssistantV2
from ibm_cloud_sdk_core.authenticators import
IAMAuthenticator

# Replace with your Watson Assistant credentials
api_key = "YOUR_API_KEY"
assistant_id = "YOUR_ASSISTANT_ID"
url = "YOUR_ASSISTANT_URL"

# Set up the Watson Assistant client
authenticator = IAMAuthenticator(api_key)
assistant = AssistantV2(
    version='2021-06-14',
    authenticator=authenticator
)
assistant.set_service_url(url)

# Define a function to interact with Watson Assistant
def chat_with_bot(user_input):
    response = assistant.message(
```

```
        assistant_id=assistant_id,
        input={
            'message_type': 'text',
            'text': user_input
        }
    ).get_result()

    # Extract the response from Watson Assistant
    bot_response = response['output']['generic'][0]['text']
    return bot_response

# Main loop for interacting with the chatbot
print("Chatbot: Hello! How can I assist you today?")
while True:
    user_input = input("You: ")

    if user_input.lower() == 'exit':
        print("Chatbot: Goodbye!")
        break

    bot_response = chat_with_bot(user_input)
    print(f"Chatbot: {bot_response}")
```