|  |  |  |
| --- | --- | --- |
|  | **MEENAKSHI SUNDARARAJAN ENGINEERING COLLEGE**  **Kodambakkam, Chennai-600024** |  |

**SB3001 – PROFESSIONAL READINESS FOR INNOVATION, EMPLOYMENT AND ENTERPRENEURSHIP**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**TOPIC: CHATBOT DEPLOYMENT IBM CLOUD USING WATSON ASSISTANT**

|  |  |
| --- | --- |
|  |  |
| **FACULTY MENTOR:** | **Vinoth Kumar. R (Assistant professor)** |
|  |  |

**Project submitted by,**

|  |  |  |
| --- | --- | --- |
| **TEAM** | **NAME** | **REGISTER NUMBER** |
| **TEAM LEADER** | *Mageswari.M* | *311521106057* |
| **TEAM MEMBER 1** | *Harini.B* | *311521106033* |
| **TEAM MEMBER 2** | *Jayashrree Priya.L* | *311521106043* |
| **TEAM MEMBER 3** | *Akshaya.E* | *311521106006* |

***Project report format***

1. **ABSTRACT**
2. **INTRODUCTION**
   1. Project Overview
   2. Purpose
3. **IDEATION AND PROPOSED SOLUTION**
   1. Problem statement definition
   2. Empathy Map Canvas
   3. Ideation and Brainstorming
   4. Proposed Solution
4. **REQUIREMENTS ANALYSIS**
   1. Functional Requirements
   2. Non-Functional Requirements
5. **PROJECT DESIGN** 
   1. Briefing
   2. Solution and Technical Architecture
   3. User Stories
6. **SOLUTIONS**
   1. Development Part I
   2. Development Part II
7. **RESULTS**

7.1 Performance Metrics

1. **ADVANTAGES AND DISADVANTAGES**
2. **CONCLUSION**
3. **FUTURE SCOPE**

SOURCE CODE

APPENDIX

**ABSTRACT**

Chatbot deployment with IBM Cloud Watson Assistant is a straightforward process that can be completed in a few simple steps. This abstract provides a high-level overview of the key steps involved in deploying a chatbot using Watson Assistant, as well as some key considerations and best practices.

The first step is to create a Watson Assistant workspace. This workspace will contain all the data and resources needed to train and deploy your chatbot, including intents, entities, dialog flows, and integrations.

Once your workspace is created, you can begin training your chatbot. This involves providing Watson Assistant with examples of the types of conversations that you want your chatbot to be able to handle. Watson Assistant will use this data to learn how to identify user intents and generate appropriate responses.

Once your chatbot is trained, you can deploy it to a variety of channels, including web chat, mobile apps, and voice assistants. Watson Assistant provides a few tools and resources to help you with this process, including code snippets and documentation.

When deploying a chatbot, it is important to consider the following factors:

• The target audience: What kind of users will be interacting with your chatbot? What are their needs and expectations?

• The channel(s) on which you will be deploying your chatbot: Each channel has its own unique capabilities and limitations. Consider which channel(s) will be the most effective for reaching your target audience.

• The integrations you will need to implement: Does your chatbot need to be integrated with any other systems, such as a CRM or customer support system? If so, you will need to implement the appropriate integrations.

• Once your chatbot is deployed, it is important to monitor its performance and adjust as needed. Watson Assistant provides a number of tools to help you with this, such as analytics and reporting capabilities.

By following the steps outlined in this abstract, you can successfully deploy a chatbot using IBM Cloud Watson Assistant. By carefully considering the factors listed above, you can ensure that your chatbot is effective and meets the needs of your target audience.

**INTRODUCTION**

Cloud application development is the process of designing, building, and deploying applications that run on cloud platforms. Cloud platforms provide developers with access to a wide range of resources, including computing power, storage, and networking, on a pay-as-you-go basis. This makes cloud application development a cost-effective and scalable way to build and deploy applications. In Addition to cloud-native technologies, cloud application developers also need to be familiar with cloud development platforms such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). These platforms provide a wide range of services that can be used to build, deploy, and manage cloud applications.

**What is a chatbot?**

A chatbot is a computer program that simulates human conversation with an end user. Though not all chatbots are equipped with artificial intelligence (AI), modern chatbots increasingly use conversational AI techniques like natural language processing (NLP) to understand the user’s questions and automate responses to them.

**How does a Chatbot work?**

This is how a typical chatbot works:

A diagram of a cloud service

Description automatically generated

The earliest chatbots were essentially interactive FAQ programs, programmed to reply to a limited set of common questions with pre-written answers. Unable to interpret natural language, they generally required users to select from simple keywords and phrases to move the conversation forward. Such rudimentary traditional chatbots are unable to process complex questions, nor answer simple questions that haven’t been predicted by developers.

Over time, chatbot algorithms became capable of more complex rules-based programming and even natural language processing, allowing customer queries to be expressed in a conversational way. This gave rise to a new type of chatbot, contextually aware and armed with machine learning to continuously optimize its ability to correctly process and predict queries through exposure to more and more human language.

Modern AI chatbots now use natural language understanding (NLU) to discern the meaning of open-ended user input, overcoming anything from typos to translation issues. Advanced AI tools then map that meaning to the specific “intent” the user wants the chatbot to act upon and use conversational AI to formulate an appropriate response. These AI technologies leverage both machine learning and deep learning—different elements of AI, with some nuanced differences—to develop an increasingly granular knowledge base of questions and responses informed by user interactions. This sophistication, drawing upon recent advancements in large language models (LLMs), has led to increased customer satisfaction and more versatile chatbot applications.

***Project Overview:***

This project is entirely going to help us with using an IBM cloud chatbot with a Watson assistant and people can use this chatbot to know their specific needs.

***Purpose:***

Chatbots can make it easy for users to find information by instantaneously responding to questions and requests—through text input, audio input, or both—without the need for human intervention or manual research.

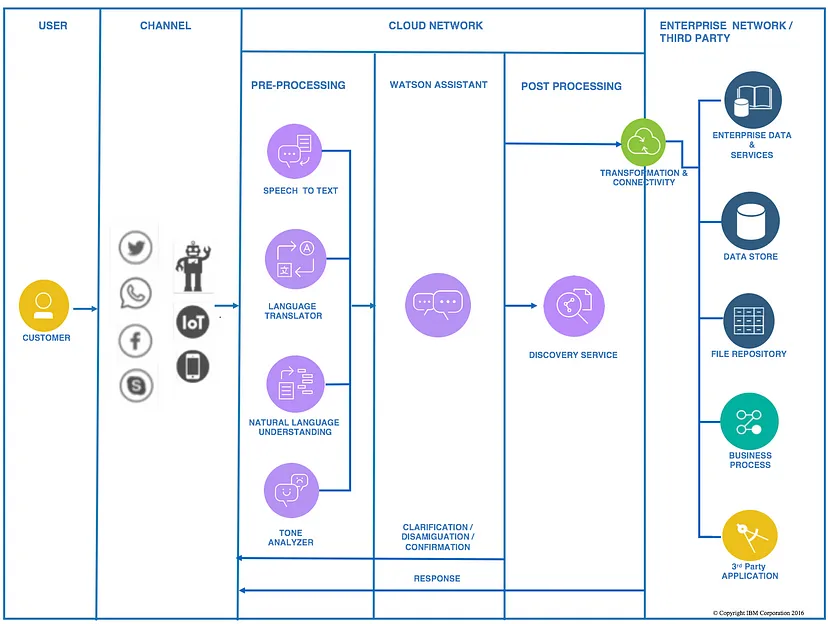
Chatbot technology is now commonplace, found everywhere from smart speakers at home to consumer-facing instances of SMS, WhatsApp, and Facebook Messenger to workplace messaging applications like Slack. The latest evolution of AI chatbots, often referred to as “intelligent virtual assistants” or “virtual agents,” can not only understand free-flowing conversation through use of sophisticated language models but even automate relevant tasks. Alongside well-known consumer-facing intelligent virtual assistants like Apple's Siri and Amazon Alexa, virtual agents are also increasingly used to in an enterprise context to assist customers and employees.

**IDEATION AND PROPOSED SOLUTION**

***Problem Statement***

Create a helpful virtual guide using IBM Cloud Watson Assistant. Customize the chatbot to assist users on popular messaging platforms like Facebook Messenger and Slack. Provide useful information, answer FAQs, and offer a friendly conversational experience. Empower users with quick access to information and create meaningful connections through your virtual guide!

***Empathy Map:***

******

***Ideation and Brainstorming:***

Some of the ideas for how an IBM Cloud chatbot using Watson Assistant could be used to help people:

1. *Customer service*: Chatbots can be used to provide customer service 24/7, answering questions, resolving issues, and providing support to customers.
2. *Sales and marketing:* Chatbots can be used to generate leads, qualify prospects, and answer sales questions. They can also be used to provide product demos and customer testimonials.
3. *Education and training:* Chatbots can be used to provide educational content and training to students and employees. They can also be used to help people learn new skills and knowledge.
4. *Productivity and efficiency:* Chatbots can be used to automate tasks, manage calendars and to-do lists, and provide reminders. They can also be used to help people stay organized and on top of their work.
5. *Entertainment and leisure*: Chatbots can be used to provide entertainment and leisure activities, such as playing games, telling stories, and generating jokes. They can also be used to help people find and book activities and events.

Here are some specific examples of IBM Cloud chatbots using Watson Assistant that could be developed:

1. A customer service chatbot for a bank that could answer questions about accounts, transactions, and fees. It could also help customers resolve issues and file claims.
2. A sales chatbot for an e-commerce company that could help customers find products, compare prices, and place orders. It could also provide personalized recommendations and answer questions about shipping and returns.
3. An educational chatbot for a university that could provide students with access to course materials, answer questions about assignments, and schedule appointments with professors.
4. A productivity chatbot for a business that could help employees manage their calendars and to-do lists, send emails and messages, and book meetings and travel.
5. An entertainment chatbot that could play games, tell stories, and generate jokes. It could also help people find and book activities and events.

These are just a few ideas, and the possibilities are endless. IBM Cloud Watson Assistant is a powerful tool that can be used to build chatbots that can help people in a variety of ways.

***Proposed Solution:***

The proposed solution of this project will be a chatbot which can be designed in the following way:

**Project Steps**

**Phase 1: Problem Definition and Design Thinking**

**Problem Definition:** The project involves creating a chatbot using IBM Cloud Watson Assistant. The goal is to develop a virtual guide that assists users on messaging platforms like Facebook Messenger and Slack. The chatbot should provide helpful information, answer frequently asked questions (FAQs), and offer a friendly conversational experience. The project includes designing the chatbot's persona, configuring responses, integrating with messaging platforms, and ensuring a seamless user experience.

**Design Thinking:**

1. **Persona Design:** Define the chatbot's persona, including its name, tone, and style of communication.
2. **User Scenarios**: Identify common user scenarios and FAQs that the chatbot should be able to address.
3. **Conversation Flow:** Design the conversation flow, outlining how the chatbot responds to user queries and prompts.
4. **Response Configuration:** Configure the chatbot's responses using Watson Assistant's intents, entities, and dialog nodes.
5. **Platform Integration:** Integrate the chatbot with popular messaging platforms like Facebook Messenger and Slack.
6. **User Experience**: Ensure a seamless and user-friendly experience, with clear prompts and informative responses.

**Phase 2: Innovation**

Consider implementing advanced features such as natural language understanding (NLU) for more accurate user intent recognition.

**Phase 3: Development Part 1**

Start building the chatbot using IBM Cloud Watson Assistant.

**Phase 4: Development Part 2**

Continue building the chatbot by integrating it with messaging platforms and refining the responses.

**Phase 5: Project Documentation & Submission**

Document the chatbot deployment project and prepare it for submission.

**Documentation**

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

**Submission**

1. Share the GitHub repository link containing the project's code and files.
2. Provide instructions on deploying and interacting with the chatbot on messaging platforms.
3. Write a detailed README file explaining how to navigate the website, update content, and any dependencies.

The Chatbot that we are going to design on:

*TOURISM*

A tourism-based chatbot using IBM Cloud and Watson Assistant is a computer program that can simulate conversation with humans, specifically about travels and tourism . It can be used to provide information about locations, such as common places, information, and other recommendations. It can also be used to take food orders and make reservations.

To build a tourism-based chatbot using IBM Cloud and Watson Assistant, you will need to:

* A tourism-based chatbot using IBM Cloud and Watson Assistant can be a valuable tool for tourism businesses of all sizes. It can help to improve customer service, increase engagement, and generate leads and sales.
* Improved customer service: A chatbot can provide 24/7 customer support, answering questions about destinations, attractions, events, and more. It can also help customers to book travel arrangements, purchase tickets, and make reservations.
* Increased engagement: A chatbot can help to increase engagement with tourists by providing them with personalized recommendations, sharing interesting facts about the destination, and running contests and promotions.
* Generated leads and sales: A chatbot can help to generate leads and sales by collecting contact information from potential customers and qualifying them for sales calls or email marketing campaigns.

Here are some examples of how a tourism-based chatbot could be used:

* A chatbot could help tourists to plan their trip by providing them with information about different destinations, attractions, and events. It could also help them to book travel arrangements, purchase tickets, and make reservations.
* A chatbot could help tourists to get around their destination by providing them with information about public transportation, maps, and directions. It could also help them to find restaurants, shops, and other businesses.
* A chatbot could help tourists to learn about the culture and history of their destination by providing them with information about local attractions, museums, and historical sites. It could also recommend activities and events that are relevant to their interests.
* Overall, a tourism-based chatbot using IBM Cloud and Watson Assistant can be a valuable tool for tourism businesses of all sizes. It can help to improve customer service, increase engagement, and generate leads and sales.

**REQUIREMENT ANALYSIS**

***Functional Requirements***

|  |  |  |
| --- | --- | --- |
| **S.No** | **Requirement** | **Description** |
| FR1 | Understanding user intent | Watson Assistant must be able to correctly identify the user's intent, even if it is expressed in a variety of ways. This requires Watson Assistant to have a deep understanding of natural language and the specific domain of the chatbot. |
| FR2 | Providing accurate and relevant responses | Once Watson Assistant has identified the user's intent, it must be able to provide an accurate and relevant response. This may involve retrieving information from a backend system, generating creative text formats, or answering questions in a comprehensive and informative way. |
| FR3 | Maintaining a natural and engaging conversation | Watson Assistant should be able to maintain a natural and engaging conversation with the user. This means being able to follow up on questions, provide context-aware responses, and use humor and personality (where appropriate). |
| FR4 | Learning and improving over time | Watson Assistant should be able to learn from its interactions with users and improve its performance over time. This may involve learning new intents, entities, and dialog flows, or improving its ability to understand and respond to complex questions. |

***Non-Functional Requirements***

|  |  |  |
| --- | --- | --- |
| **S.No** | **Requirements** | **Description** |
| NFR1 | Scalability | The chatbot must be able to handle a large number of users and conversations simultaneously. |
| NFR2 | Security | The chatbot must be able to protect user data from unauthorized access. |
| NFR3 | Reliability | The chatbot must be available and reliable 24/7. |
| NFR4 | Performance | The chatbot must be able to respond to user queries quickly and efficiently. |
| NFR5 | Usability | The chatbot must be easy to use for both users and administrators. |
| NFR6 | Maintainability | The chatbot must be easy to maintain and update. |
| NFR7 | Accessibility | The chatbot must be accessible to users with disabilities. |
| NFR8 | Compliance | The chatbot must comply with all relevant laws and regulations. |

In addition to these general non-functional requirements, Watson Assistant chatbots may also have specific non-functional requirements depending on the industry and use case. For example, a chatbot used in healthcare may need to meet additional HIPAA compliance requirements.

**PROJECT DESIGN**

***Briefing:***

To deploy this chatbot, you will need to:

1. Create a Watson Assistant service instance on IBM Cloud.

2. Obtain a Watson Assistant API key.

3. Update the watson\_assistant\_url and watson\_assistant\_api\_key variables in the source code with your own values.

4. Save the source code as a Python file (e.g. tourism\_bot.py).

5. Run the following command to deploy the chatbot:

6. python tourism\_bot.py

7. This will start a conversation with the chatbot. You can type in messages and the chatbot will respond.

Here are some additional tips for deploying a chatbot with IBM Cloud Watson Assistant:

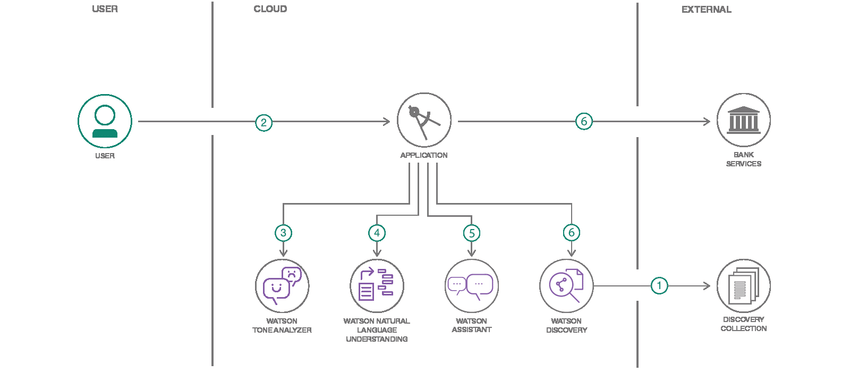
• You can use the Watson Assistant web UI to train your chatbot on your data.

• You can use Watson Assistant integrations to connect your chatbot to other services, such as Slack or Facebook Messenger.

• You can monitor the performance of your chatbot using the Watson Assistant analytics dashboard.

Consider implementing advanced features such as natural language understanding (NLU) for more accurate user intent recognition.

***Solution and Technical Architecture***

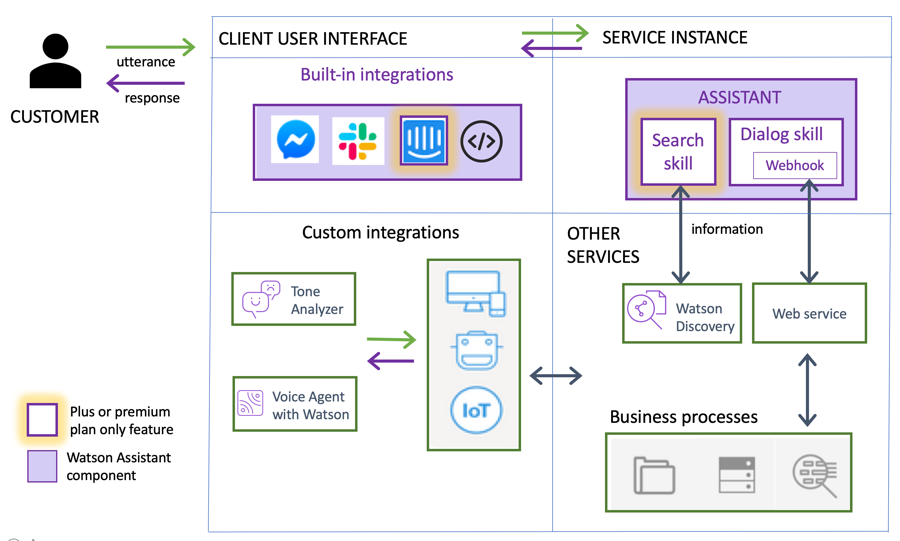


The following figure shows Watsonx Assistant architecture that's common for all implementations. In this architecture, Users interact with the assistant through one or more of these integration points:

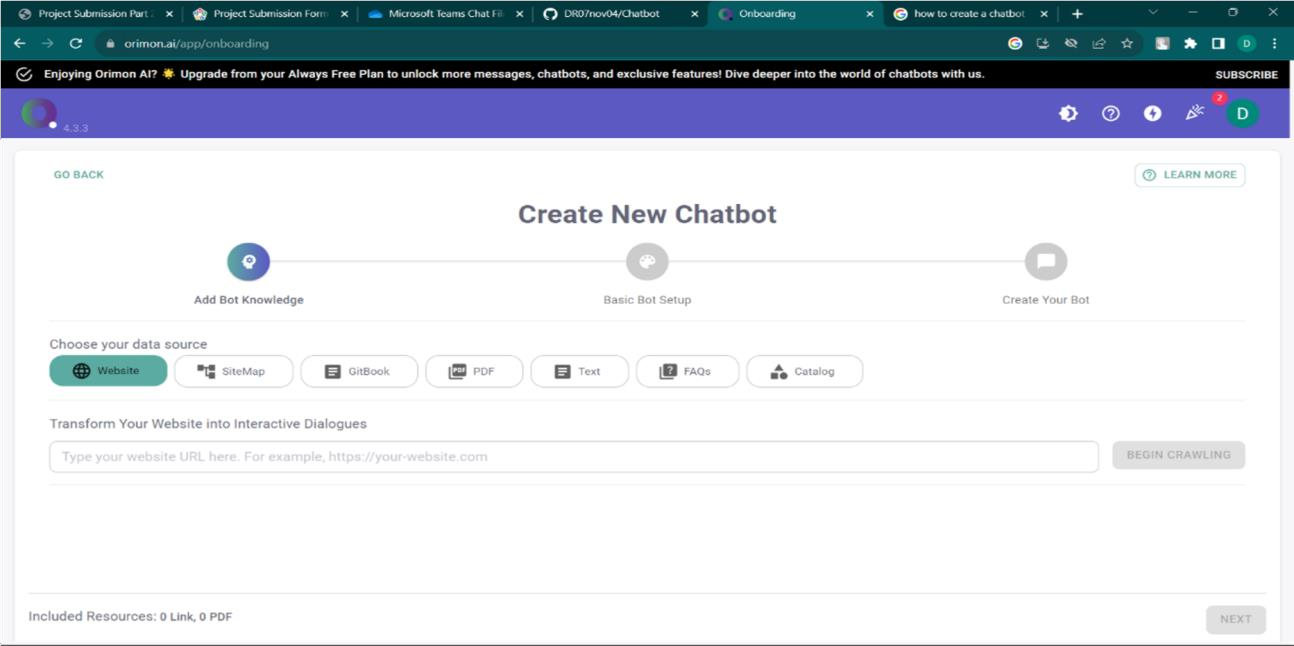
* + A web chat code snippet that runs in your existing website
  + A virtual assistant that you publish directly to an existing third-party messaging platform, such as WhatsApp with Twilio, Slack, or Facebook Messenger
  + A voice assistant that the user accesses over the phone
  + A custom application that you develop, such as a mobile app or a robot with a voice interface.

The assistant receives user input and routes it to the dialog skill.

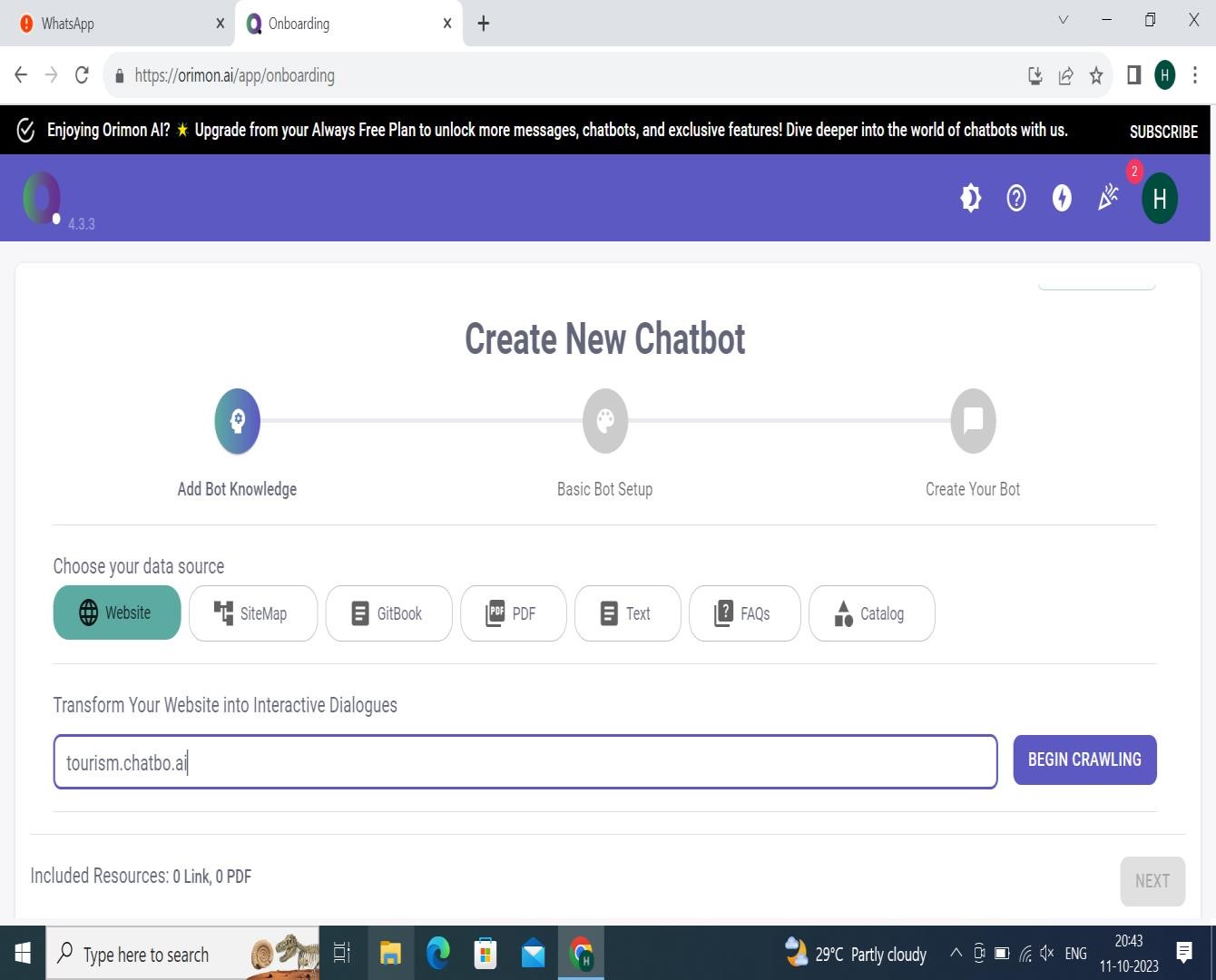
* The dialog skill interprets the user input further, then directs the flow of the conversation. The dialog gathers any information it needs to respond or perform a transaction on the user's behalf.
* Any questions that cannot be answered by the dialog skill are sent to the search skill, which finds relevant answers by searching the company knowledge bases that you configure for the purpose.

****A typical approach used when deploying watsonx Assistant.

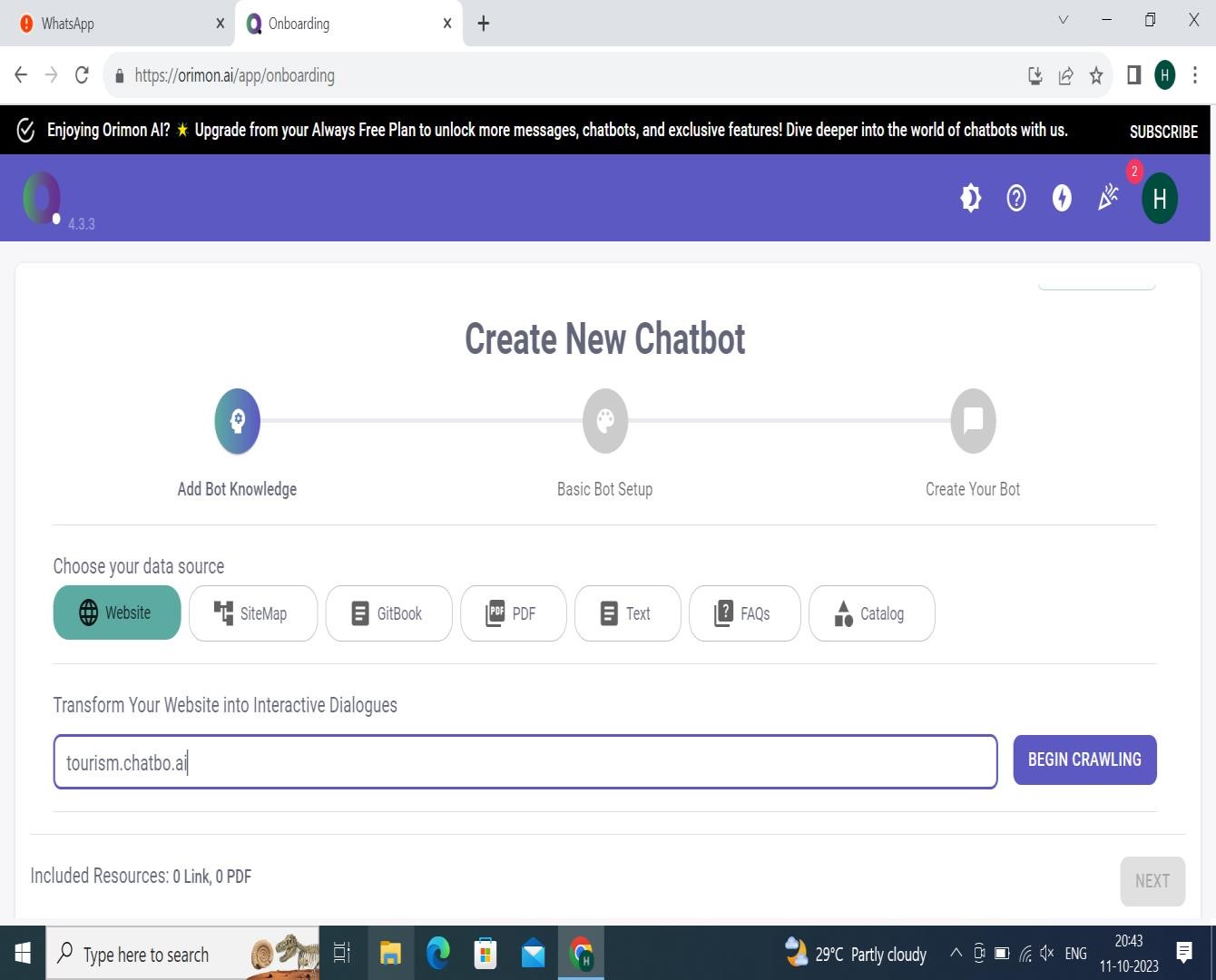
**Step 1: Create a New Chatbot**



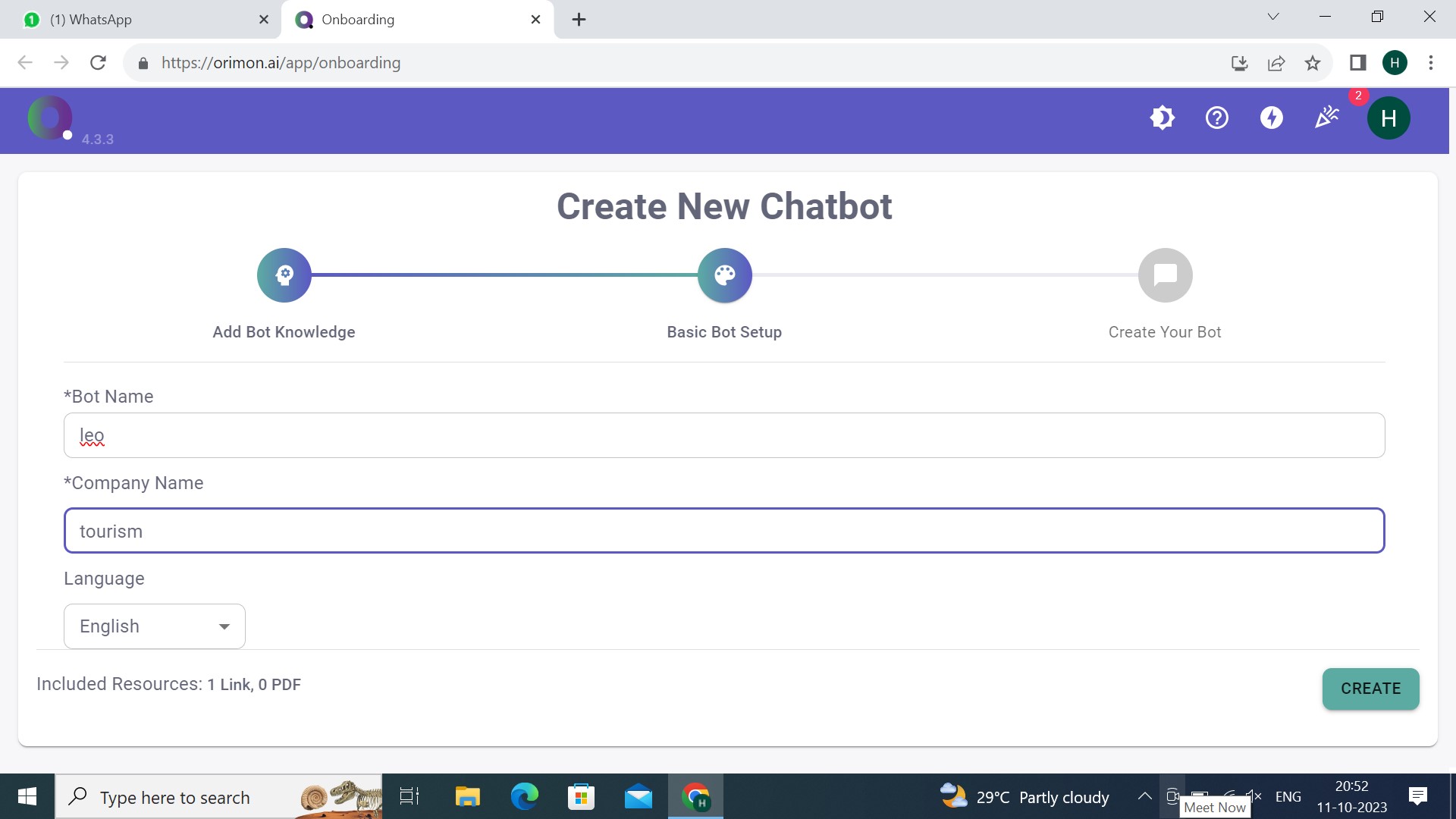
**Step 2: Name your Chatbot and frame a URL**



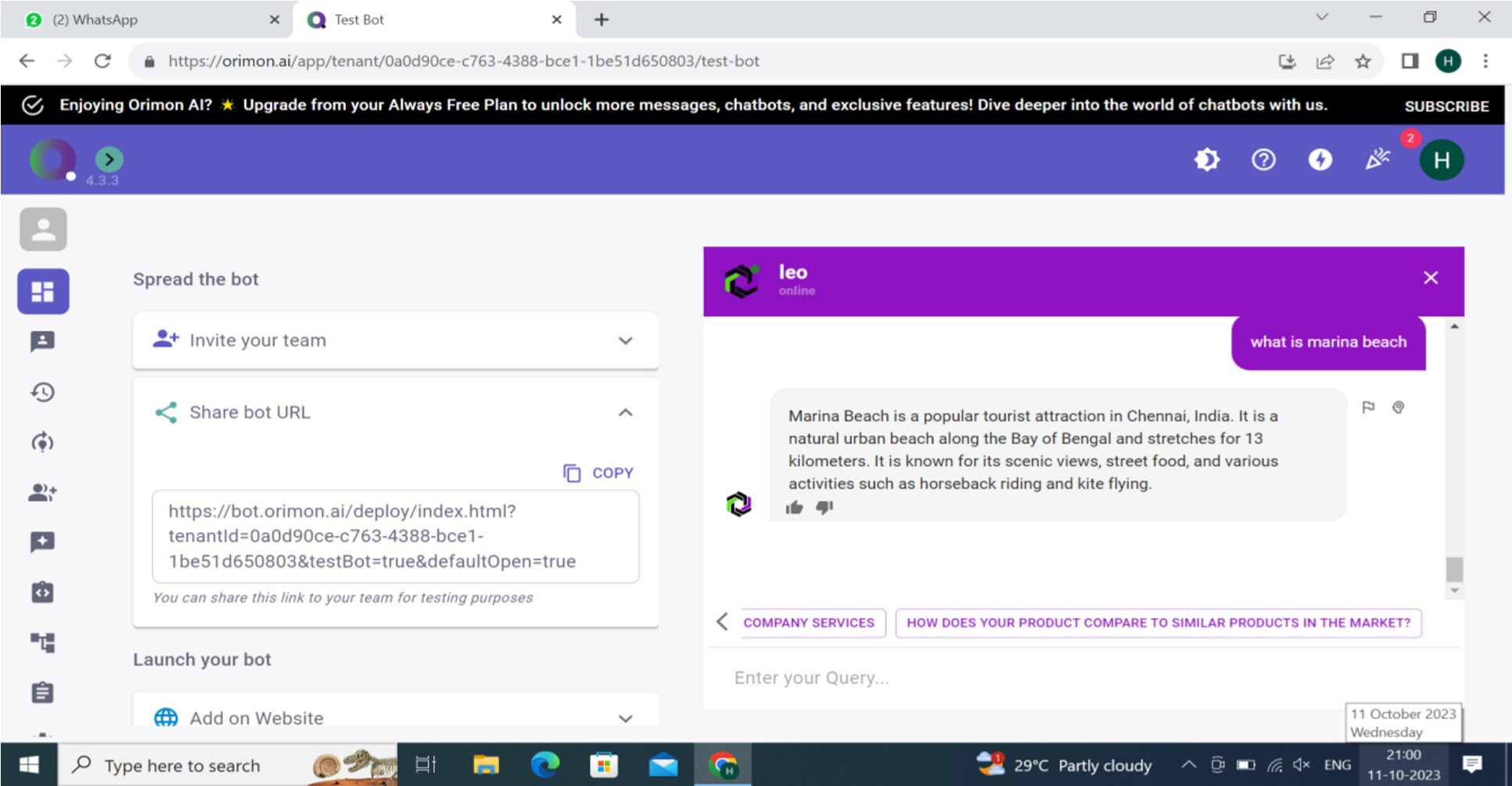
**Step 3: Select your framed link and create a website**



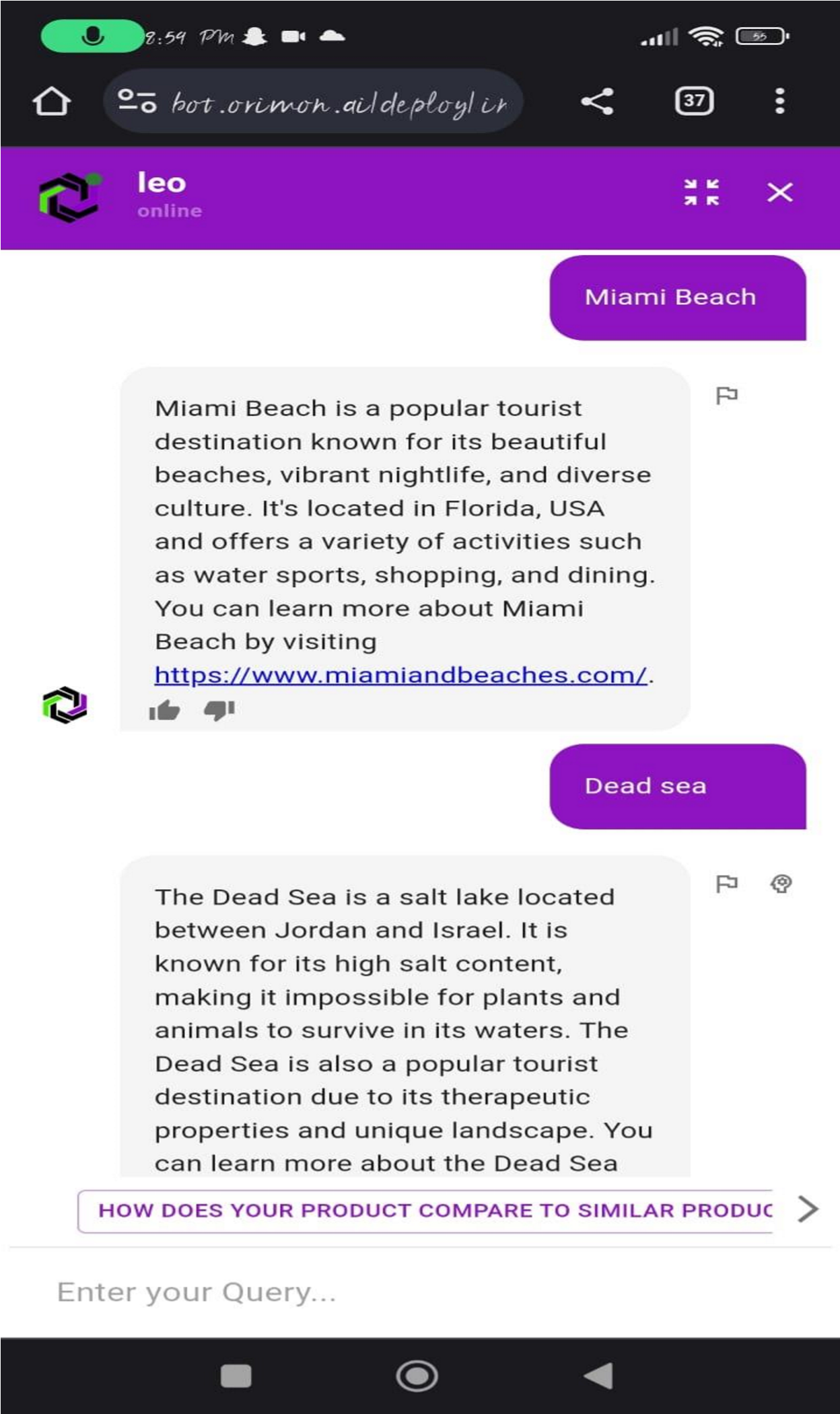
**Step 4: Setup your Bot**



**Step 5: Test your AI and set it.**



***User Stories :***



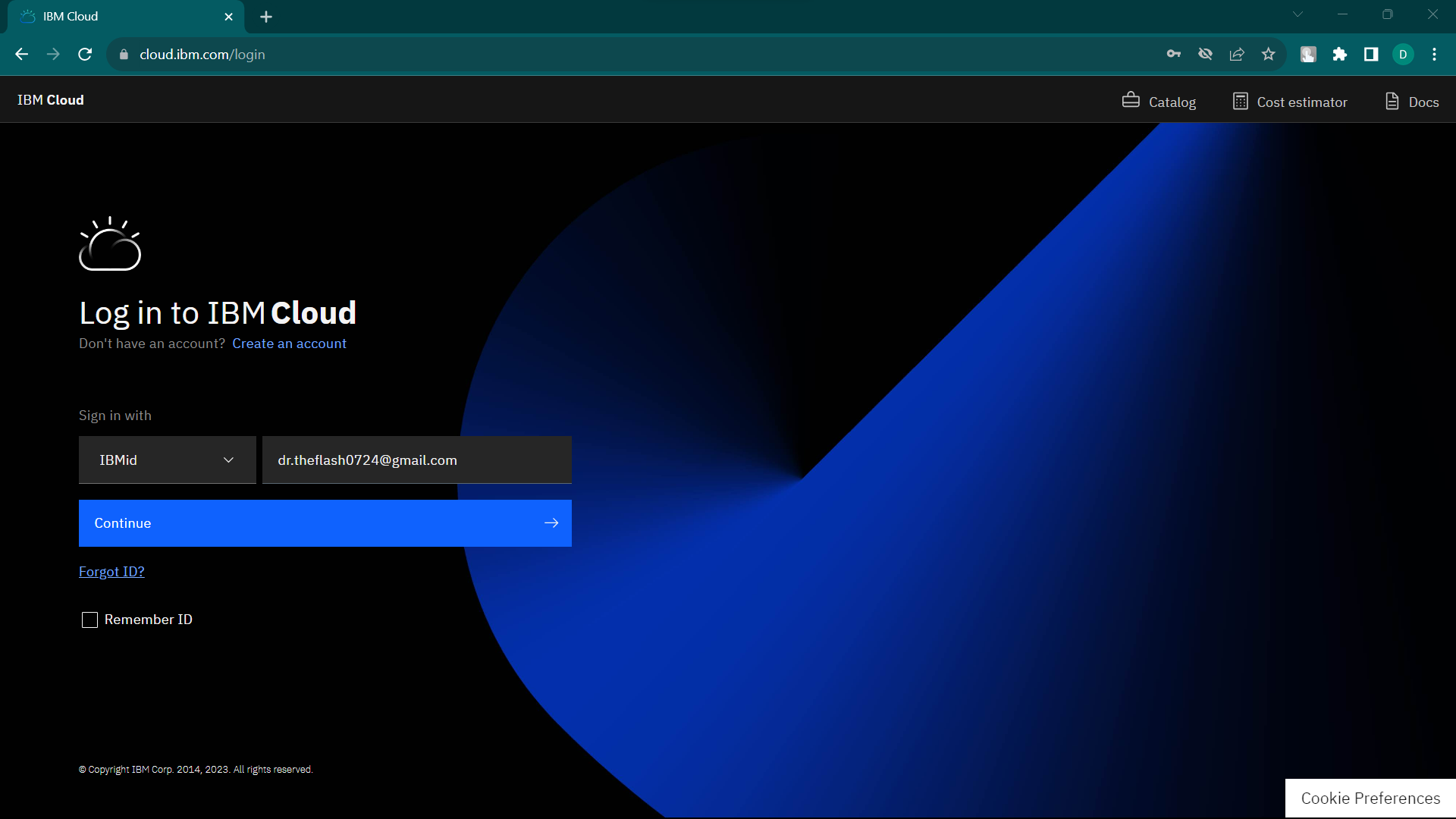
**AI link/URL:**

https://bot.orimon.ai/deploy/index.html?tenantId=0a0d90ce-c763-4388-bce11be51d650803&testBot=true&defaultOpen=true

**SOLUTION**

***Development: Part 1***

Task 1: Create the Assistant service



1. The first task is to create an instance of Watson Assistant on IBM Cloud.
2. Make sure that you are logged in to your IBM Cloud account. Click Catalog and then click Services > Watson > Assistant.

A screenshot of a computer

Description automatically generated

IBM Cloud Watson Services

1. For the service name, type **Tourism**. Click Create. Assistant Service will be displayed.

A screenshot of a chatbot

Description automatically generated

1. Click the Launch tool to open the Watson Assistant workspace.

Task 2: Create a workspace

1. In the Workspaces section, click Create. A workspace will be created.

A screenshot of a computer

Description automatically generated

1. Type a name for the workspace., the workspace name is **TOURISM AND TRAVELS**.

Task 3: Create Intents

Add intents. An intent is a group of examples of things that a user might say to communicate a specific goal or idea. To identify intents, start with something that a user might want and then list the ways that the user might describe it. For each intent, think of the various ways that a user might express his or her desire—those are the examples. Examples can be developed by using a crowdsourcing approach.

For example, in a discussion with the user, we might gather this set of standard questions that support received from users:

* **Which region are you from?**
* **Phone number**
* **Gmail id**
* **Name if needed.**

Each of those questions is documented as a frequently asked question in the support team's document repository. Some solutions persist in a relational database in the form of application > problem > solution.

Based on the questions, you can extract these intents:

* **Explain about this chatbot to a 5-year-old.**
* **Common locations and common food.**
* **Why is this chatbot?**

Add those intents to the workspace: From the Build page, click Intents and click Create New.

Intents

For the intent name, type applicationAccess after the number sign (#).

Questions

For each intent, add examples to train the conversation for intent recognition.

Create the Goodbyes intent and add examples for it.

To get the IT support demonstration intents, click the Import link on the Intents page to import the Intents.csv file from the GitHub repository.

Task 4: Test the intent

Next, test your conversations.

1. As soon as you create an intent, you can test it by clicking Ask Watson icon in the top, right-hand side of the conversation editor.

A screenshot of a credit card

Description automatically generated

Ask Watson

1. Enter one of the examples. You should get the #greetings intent identified by Watson. Enter other greetings to test the #greetings intent.
2. Test a few of the other intents that you imported from the .csv file.

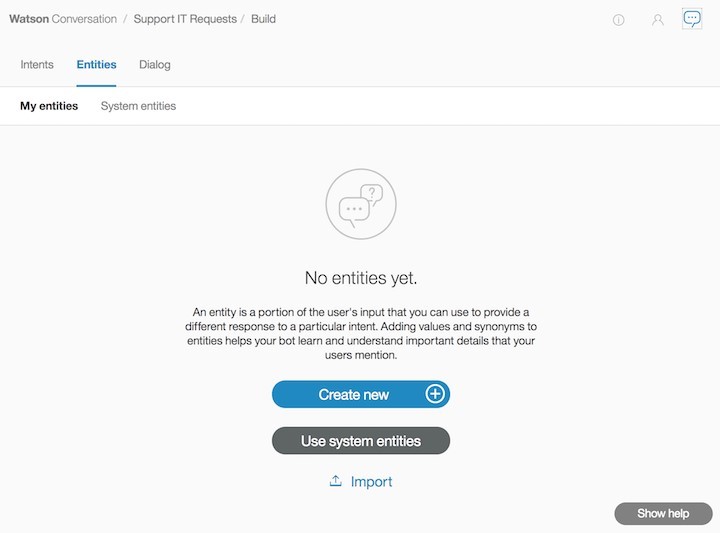
***Development: Part 2***

\*\*\*Images are subjected to the demo we’ve seen and learned, since, the cloud is not activated, we have given our idea using the demo.

# Task 5: Add Entities

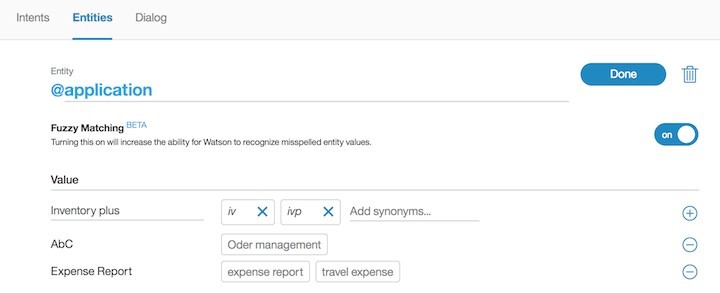
An entity is a portion of the user's input that you can use to provide a different response to a particular intent.

1. Click Entities. On the Entities page, click Create new.



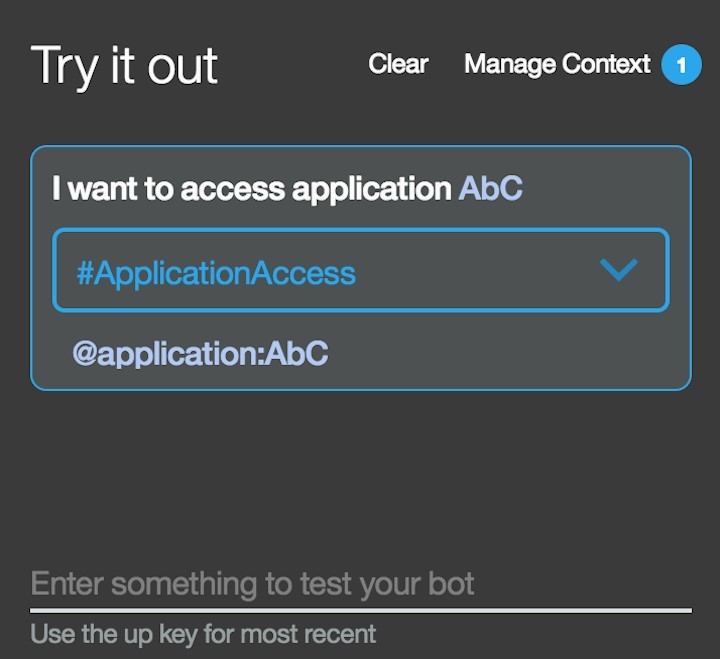
* Adding values and synonyms to entities helps your chatbot learn important details that your users might mention.
* Each entity definition includes a set of specific entity values that can be used to trigger different responses. Each value can have multiple synonyms that define different ways that the same value can be specified in user input.

2. Create entities to represent to the application what the user wants to access.



* Fuzzy logic is a feature that allows Watson Assistant to accept misspelled words. You can enable this feature at the entity level.
* As you did for intents, you can reuse entities' definitions through the export and import capabilities. Import the wcs-workspace/ITSupport-Entities.csv file.

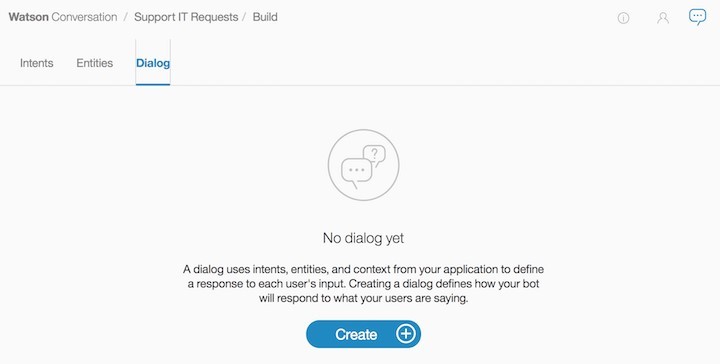
3. If you click the Ask Watson icon immediately after you import the entities, the Watson is training message is displayed. Watson Assistant classifies the entities. You can unit-test the entities by entering I want to access application AbC. The following figure shows both the intent and entity (@application:AbC) extracted by Watson Assistant:



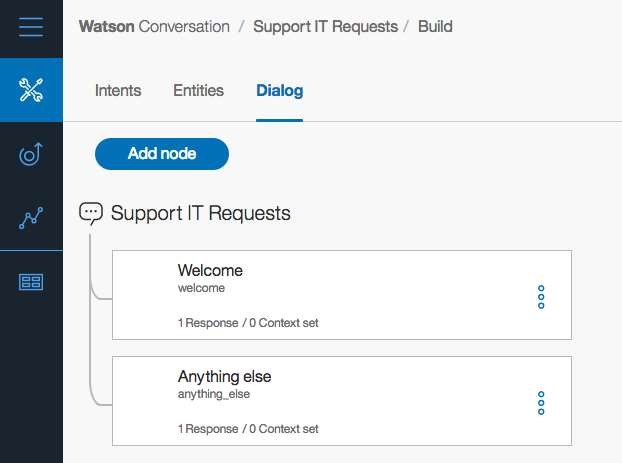
You are now ready to create the dialog flow.

# Task 6: Build the dialog flow

After you specify your intents and entities, you can construct the dialog flow.



A dialog is made up of nodes that define steps in the conversation.

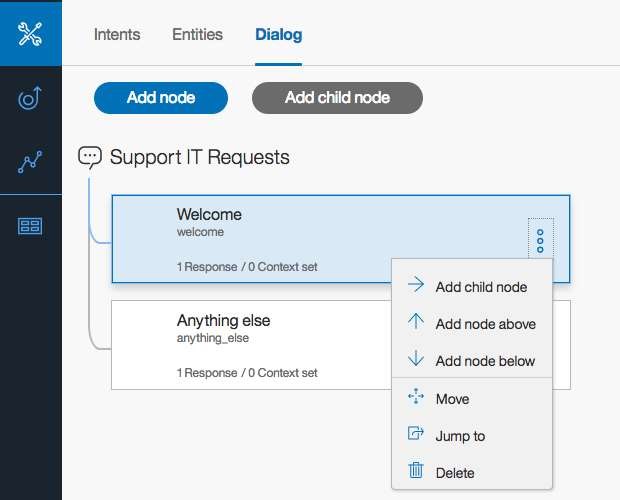


In the previous image, two dialog nodes are shown. The first node is the standard welcome message. The other node is a catch-all node named "Anything else." Dialog nodes are chained in a tree structure to create an interactive conversation with the user. The evaluation starts at the top, so the welcome node is assessed before the "Anything else" node.

If you click the welcome node, the standard Watson response is "Hello. How can I help you?" To validate how the flow works, you can click the Ask Watson icon.

Define the greetings node

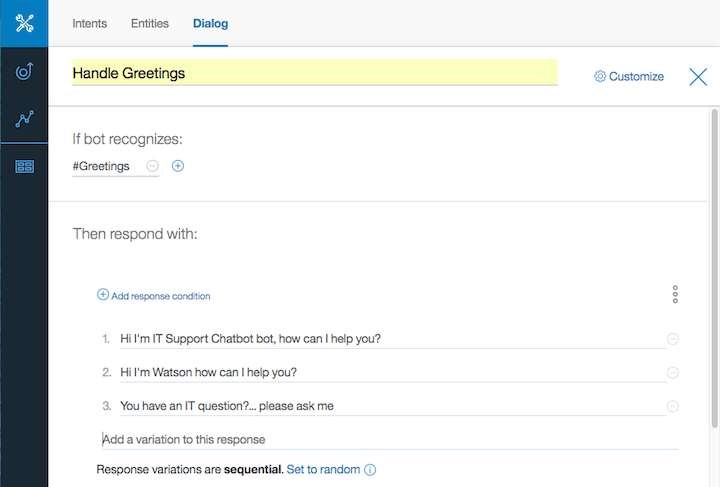
1. The first node addresses greetings in a response to a query such as "hello." Click the welcome node and click Add node below:



A new node is added between the welcome and "Anything else" nodes.

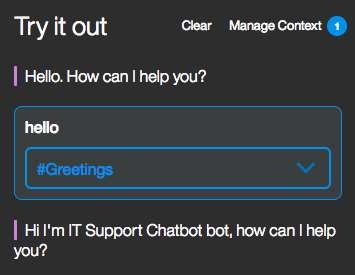
At each node level, you can expand the conversation by adding nodes. If you add nodes at the same level, the flows are parallel. Adding a child node creates a dependent track of conversation, and the conversation branches out into a tree structure.

1. Name the new node Handle Greetings. In the If bot recognizes field, change the value to #Greetings. The number sign (#) represents a prefix for intent. The condition is triggered when the Watson natural language classifier classifies the query as a greeting intent.
2. Add these responses:



The previous image also illustrates how to use the multiple responses pattern to avoid being repetitive. The bot can present different answers to the same query. You can allow the system to randomly select an answer from the list of potential responses.

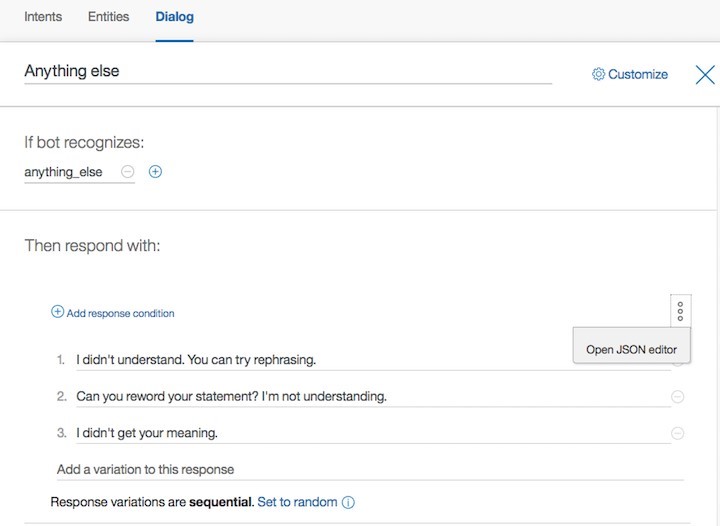
1. Unit-test your dialog by clicking the Ask Watson icon:



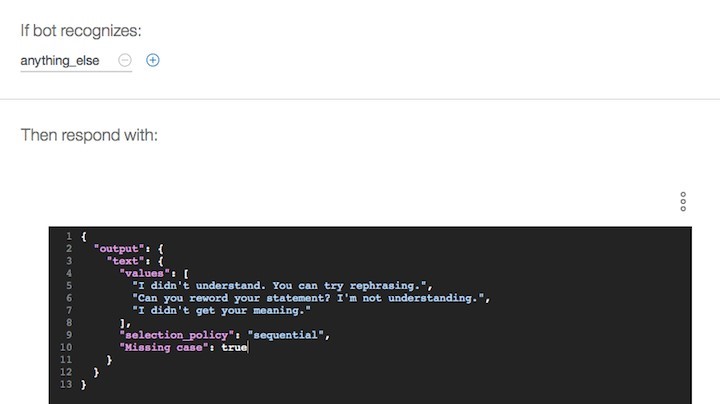
At the beginning of each conversation, the evaluation starts at the top level of dialog nodes.

Manage the "Anything else" use case

The bottom node is used when none of the defined intents are matched. It can provide a default message, as shown in this image:



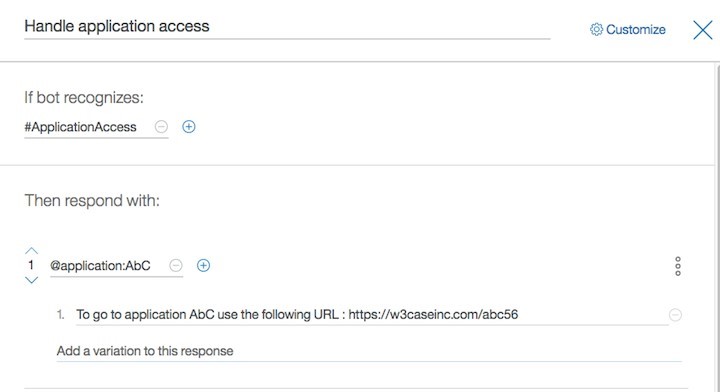
From the menu on the right side of the response area, you can open the JSON editor and assess the data that is returned as part of the conversation interaction. The JSON document includes an output JSON object with text with different values. To the output, add an attribute, name it Missing case, and set it to true. When you persist the conversation flow into a document oriented database , you can search the queries that were not addressed by the dialog nodes so that you can add more cases later, if needed.



Define the application access dialog flow

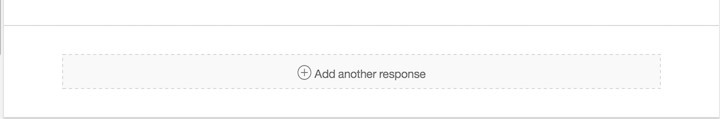
Now you can create dialog branch to handle the ApplicationAccess intent.

1. Click the "Handle Greetings" node and then click Add node. Name the new node Handle application access. In the If bot recognizes field, type #ApplicationAccess.

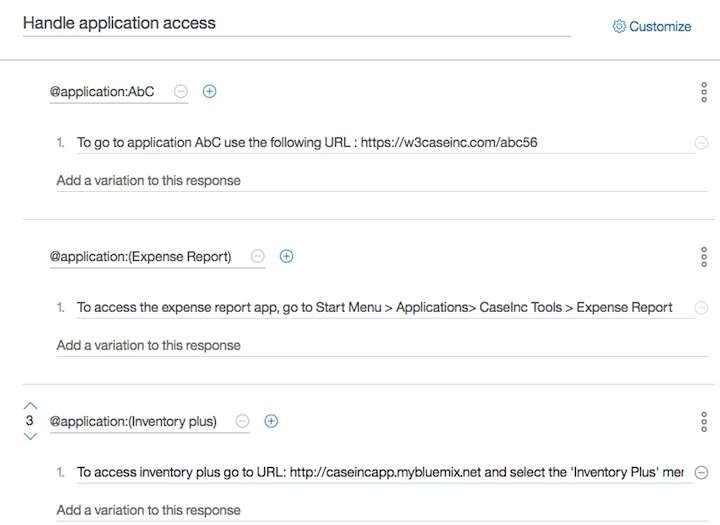


When users enter queries about application access, they will most likely specify one of the supported applications; for example, I want to access application abc. Therefore, in the Application Access node, you need to add multiple conditions by looking at the different possible entities. As illustrated in the previous image, the first condition tests on the presence of the @application:AbC entity. When it matches, the response returns a static URL. Later, you will learn how to make the URL clickable from Watson Assistant in the chatbot application.

1. To add more conditions in the same node, click Add another response.

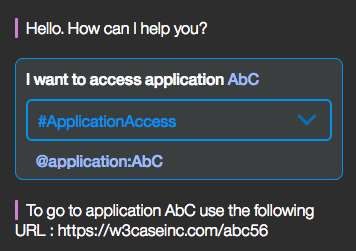


1. In the condition, click @application:ExpenseReport and provide the solution to access the app. Repeat these steps for all the entities that you defined for the application.

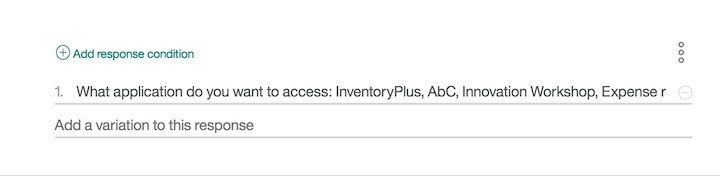


1. Test the sentence I want to access application AbC. You should get the

ApplicationAccess intent and the application AbC entity. Verify that the node match gives the expected response:



1. If the user does not specify any application, you want to ask which one to access from a list of predefined apps. This is another response pattern where predefined content can be selected by the user. Add a node as shown in this image:



1. If you enter a sentence like Access application, the expected response is proposed.
2. If you enter a choice like Inventory Plus, you might not get the expected result because the classifier might have extracted the wrong intent. From the unit test, you can enforce Watson Assistant to do a better job by selecting the right answer: select the applicationAccess intent and Watson will retrain it. Be sure to test again to ensure that the conversation works as expected.
3. Try a few statements that aren’t the same as your training data.

Adding any new entity or intent makes Watson Assistant retrain its classifier.



The evaluation round works in two stages. In the first stage, the dialog tries to find an answer in the child nodes of the contextual node. That is, the dialog tries to match all the conditions of the child nodes of the contextual node. If the final child node has a condition of "true," meaning that it is to be used if none of its siblings are matched, that node's response is processed.

If no match is found, the dialog continues to a second stage where it tries to find an answer to an input by matching the top level of dialog nodes.

The top level of dialog nodes should contain an anything\_else node as the last node, which is reached when no match occurs in the conditions of the top-level nodes. Typically, if an anything\_else node is defined, the dialog returns an answer to every user input.

At this stage, you can build a static flow. Next, you get user input and do something with it.

# Task 7: Complete advanced dialog work

Each time the dialog returns a response and waits for user input, the dialog stores the ID of the node where the conversation must resume. This node is called the contextual node. Its ID is added to the context.system.dialog\_stack property, which contains a JSON array of dialog node IDs that are on the dialog stack.

The context object.

State information for your conversation is maintained by using the context object.

The context object is a JSON object that is passed between your application and the Watson Assistant service.

This example shows context that is returned from an interaction that reaches a top node:

json

{

"context": {

"conversation\_id": "Conversation-ut",

"system": {

"dialog\_stack": [

{

"dialog\_node": "root"

}

],

"dialog\_turn\_counter": 1,

"dialog\_request\_counter": 1,

"\_node\_output\_map": {

"AbC access": [0]

},

"dialog\_in\_progress": false

}

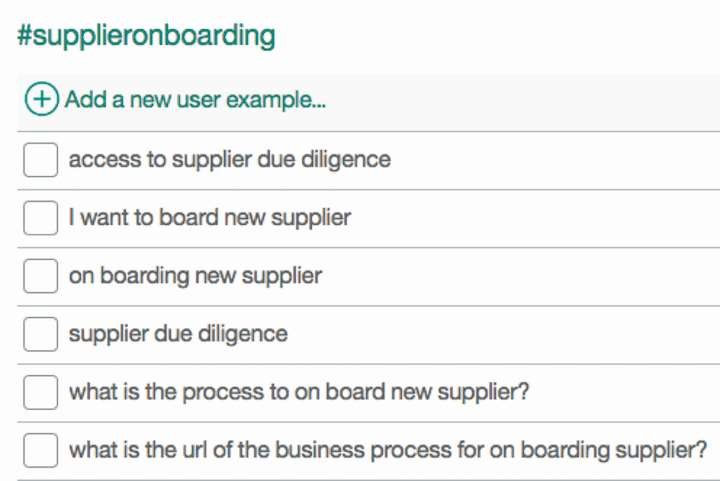
}

}

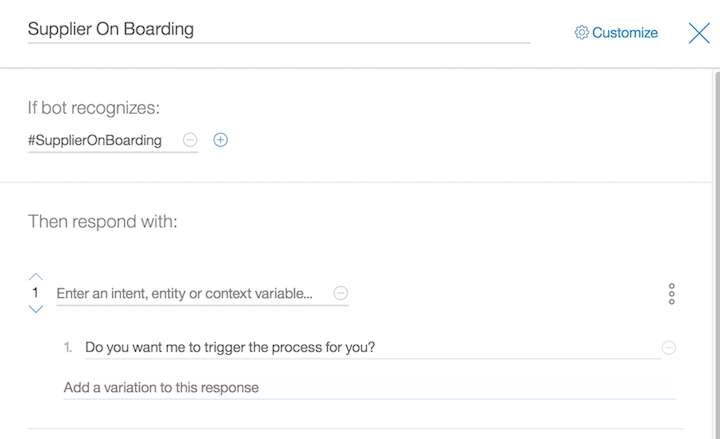
Add variables to context

Suppose that you want to return an actionable URL, meaning that the response includes a URL variable that the user can click to go to a new page. To try this scenario, you present the URL of a business process that is deployed on IBM BPM on Cloud.

1. If you didn't import the intents definition from the lab CSV file, add an intent to support the user's query about accessing the "Supplier on boarding business process."



1. Add a node to the dialog flow under the "Application access" node. Name the node Handling supplier on boarding and for its intent, type #SupplierOnBoarding.
2. In the response, access the Advanced editor to edit the JSON response object:



The response object includes an output object with the text to present and a context object. The context object has a new url variable to give IBM BPM on Cloud access and an action variable to control the behavior of the broker code. When the user enters a question like "I want to access the supplier on boarding business process,” the returned object looks like this example:

json

{"intents": [

{

"intent": "supplieronboarding",

"confidence": 0.940047025680542

}

],

"entities": [

{

"entity": "supplier",

"value": "supplier"

}

],

"input": {

"text": " I want to access the supplier on boarding business process" },

"output": {

"log\_messages": [],

"text": [

"To access the Supplier On Boarding business process use the url: "

],

"nodes\_visited": [

"Supplier On Boarding",

"Provide URL for Supplier on boarding"

]

},

"context": {

"conversation\_id": "Conversation-ut",

"system": {

"dialog\_stack": [

{

"dialog\_node": "root"

}

],

"dialog\_turn\_counter": 1,

"dialog\_request\_counter": 1

},

"url":

"https://vhost001.bpm.ibmcloud.com/bpm/dev/ProcessPortal/dashboards/SYSRP/RESPONS IVE\_WORK",

"action":"click"

}

}

The code that calls the Watson Assistant API can take the URL value and create a hyperlink in HTML so that the display has an actionable link:

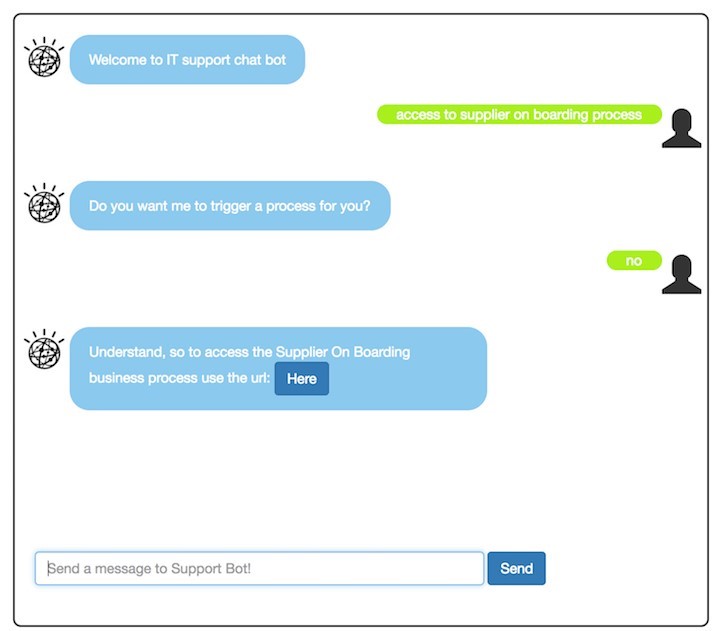
javascript if (rep.context.url != undefined) { if (rep.context.action === "click") {

rep.text=rep.output.text[0] + "<a class=\"btn btn-primary\" href=\""+rep.context.url+"\">Here</a>"

}

}

The actionable link is shown in the last interaction, with the clickable button added to the message:



Much can be done on the context object. For example, the application code can add elements in the context object before it calls the Watson Assistant service. Then, at the condition level in a node, tests can be done on those elements.

To use a context variable in your condition, use one of these formats:

* $variable\_name:value
* $variable\_name == 'value'.

The value of the condition might have been set by the application or in the response portion of a previous dialog node. In the next test, the canAccessSOD context variable is a Boolean that is set by accessing an internal authorization service that, for example, returns true if a user ID can access an application.

Use slots

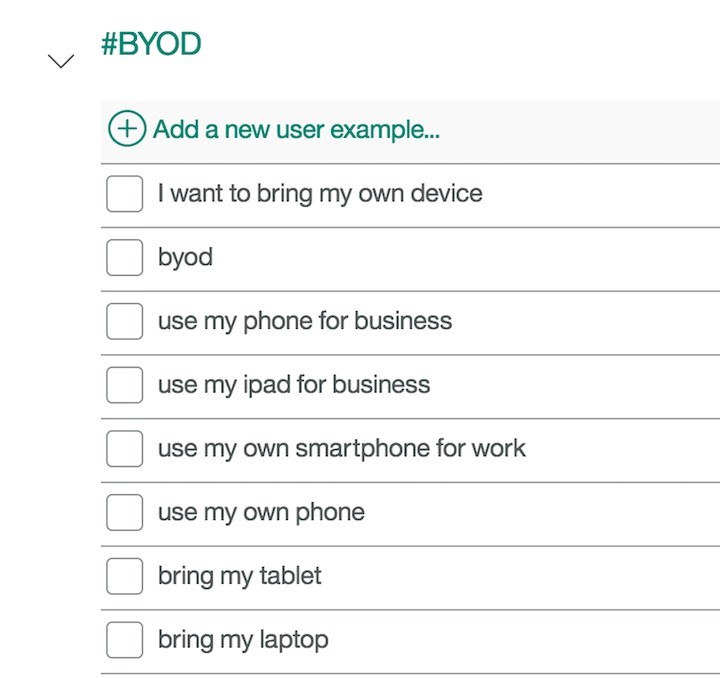
Add a dialog flow to address when a user wants to bring his or her own device. Only certain brands and devices are supported, so you must determine the brand and type of device. The goal is illustrated by this dialog:



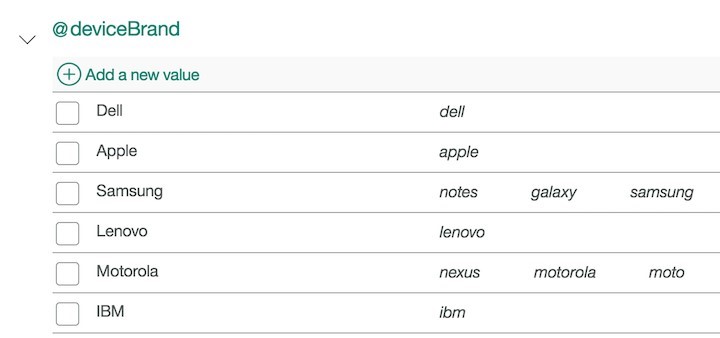
From the first query, "I want to bring my phone," Watson Assistant can get the #BYOD intent and the @deviceType:phone entity. The dialog flow asks for the brand of the device. If the device type is not extracted, it asks a question about the type of device that the user wants to bring.

One way to support this combination is to add a hierarchy of nodes and code the conditions on entity. In releases from August 2017 and later, you can use slots to get information so that you can accurately respond to the user.

1. If you didn't import the intent and entities, create an entity for the "bring your own device" question.



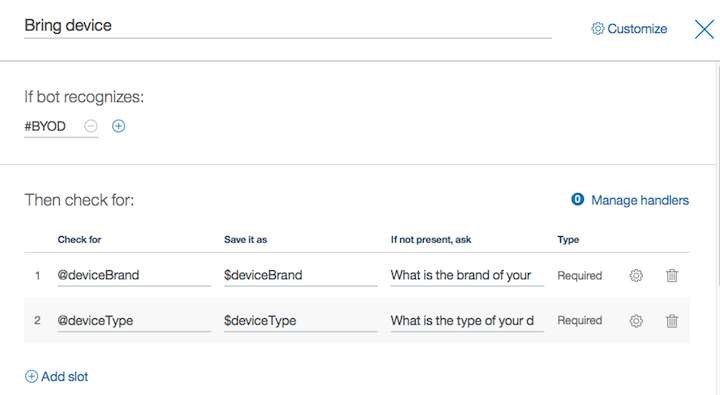
1. Create the @deviceBrand and @deviceType entities.



1. Add a flow by adding a top-level node that uses the #BYOD intent as the recognize condition. Click the Customize menu to enable slots for this node.



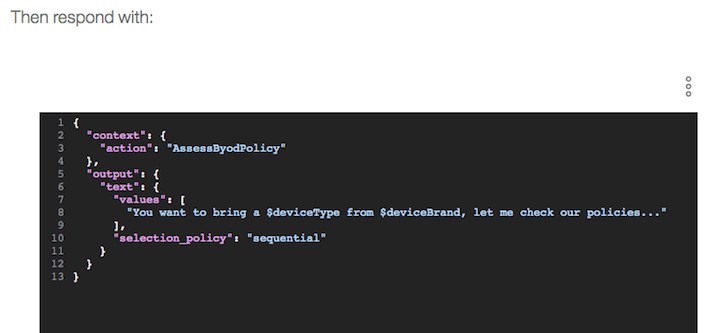
Verify that the condition changes to "Then check for."



1. Add two slots:
   * Check for the @deviceBrand entity and save the result in the $deviceBrand context variable. If the brand is not present, ask "What is the brand of your device?"
   * Check for the @deviceType entity and save the result in the $deviceType context variable. If the type is not present, ask "What is the type of your device (tablet, smartphone, computer)?"

Slots make it possible for the service to answer follow-up questions without reestablishing the user's goal.

1. For the response, use the advanced dialog to enter output text to display the content of the device type and brand:



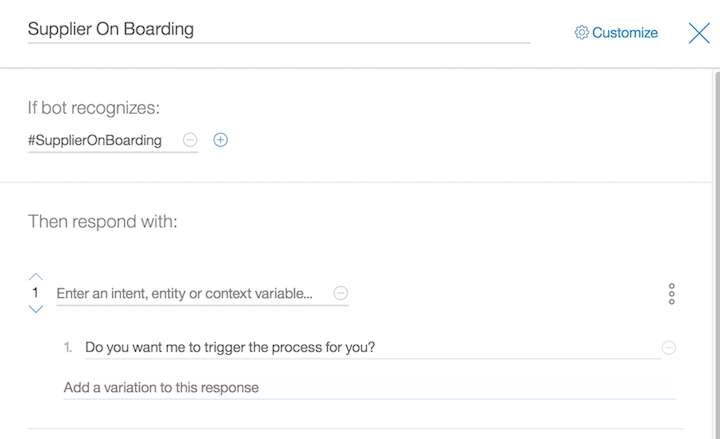
In the and then action section, be sure to set wait for user input.

Use an interaction to get parameters to call a web service.

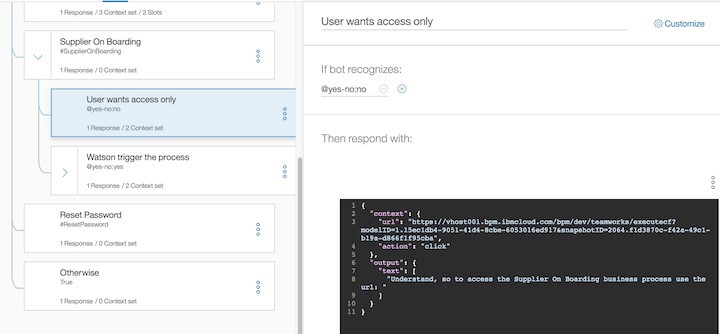
The last example is more complex, but it represents a common pattern: in the context of a dialog flow, the set of interactions aim to gather input parameters so that after the interaction is done, the broker code can call a web service and pass the parameters as part of the payload.

In this example, the IBM BPM supplier onboarding process is triggered through a SOAP call. Two input parameters exist: the company name and the product name. The client code is in the conversation broker code as a feature in the server/routes/features/supplier-bpmclient.js file.

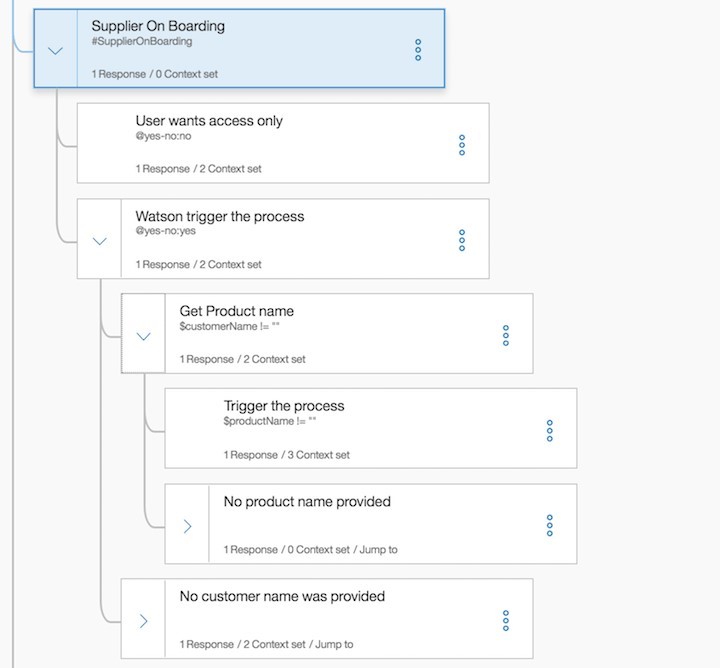
Because the focus is on the dialog, you must modify the previous flow to handle the "Supplier process" intent. The first node asks whether the user wants the chatbot to trigger the process.



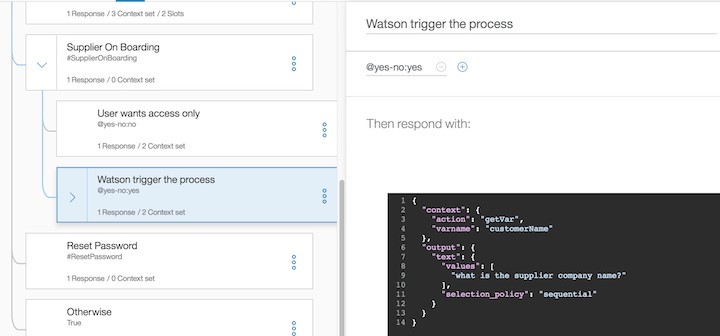
When the response is no, you use the previous behavior by providing the URL to the IBM BPM Process portal so that the user can manually start the process.



When the response is yes, the two next interactions are used to get the company name and product name. You can try to reproduce this structure:



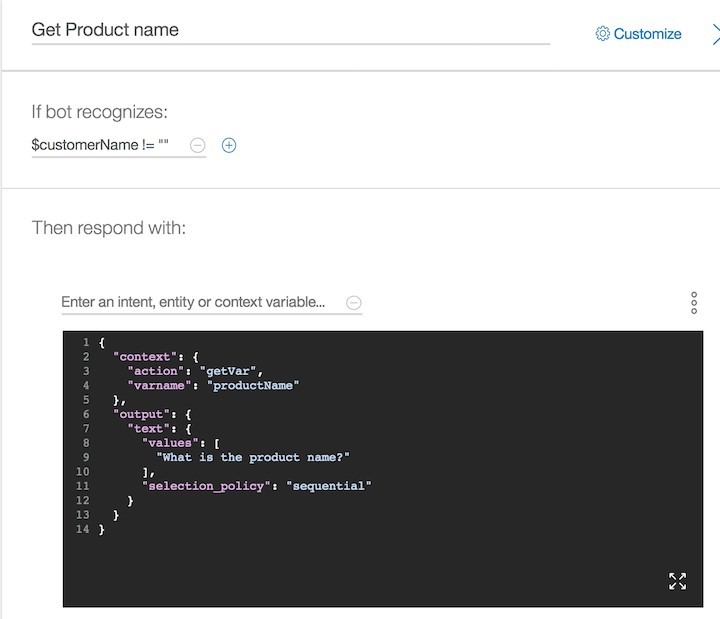
Outside of the output.text file, which provides a question about the company, the context uses two variables to drive the broker code. The action is set to getVar and the varname is set to the name of the variable to add to the context via code. On a yes response, the bot asks about the supplier company name.



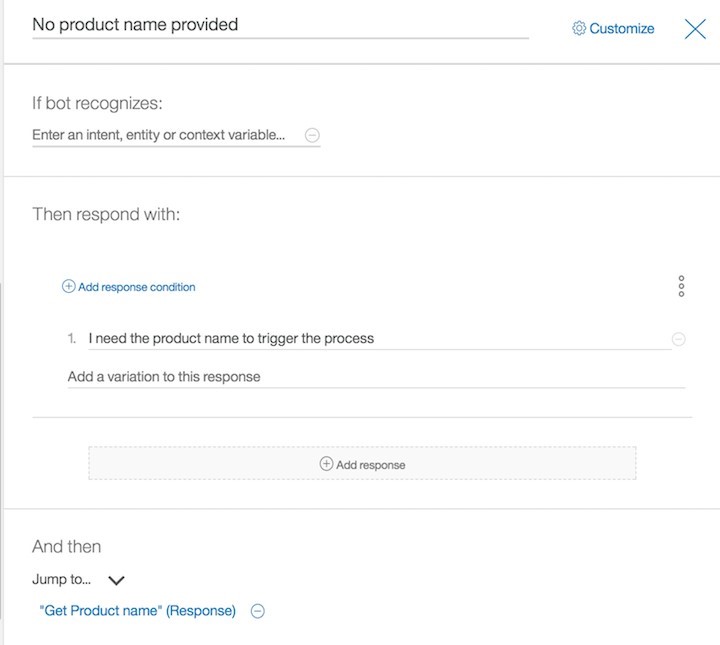
The broker code dynamically adds the specified variable in the context to keep the data for the next interaction. The following code is in the function to manage the user's response before it calls the Watson Assistant service: javascript if (req.body.context.action === "getVar") { req.body.context[req.body.context.varname] = req.body.text;

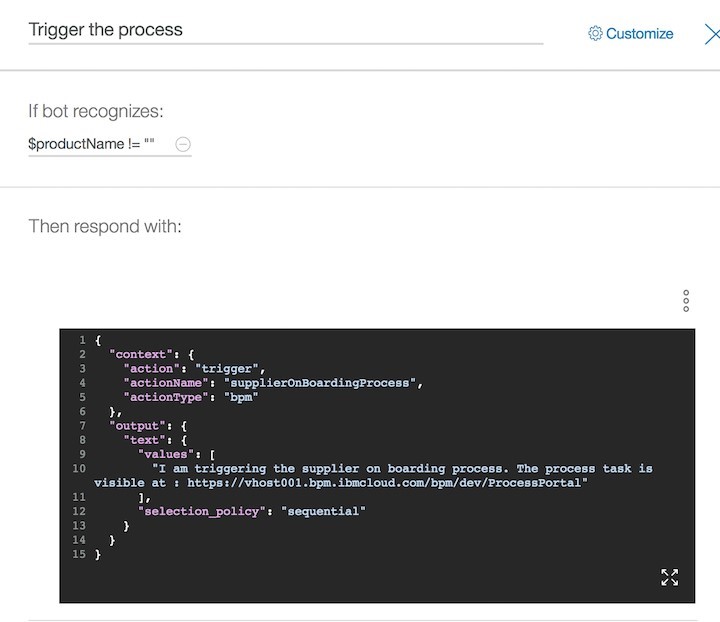
}

The same approach is done for the product name:



During the dialog execution, if the user doesn't specify the product or company name, a node in the dialog flow identifies the missing parameters. The node then routes the flow back to the node, asking for the missing parameters. As you can see in the next image, the "Jump to..." constraint is added to the node definition so that the dialog flow returns to the "Get Product name" node.



When all the parameters are set, the interaction sets the action to trigger so that the broker call can perform the SOAP call to the service. 

# Task 8: Use the API

To use the API, you need the service credentials and the tool to perform an HTTP request.

For detailed instructions, see Use Watson Assistant API.

**RESULTS**

****

Leo is a Tourism -based chatbot built on IBM Cloud Chatbot Watson Assistant. It can answer questions about places, cities, monuments as well as provide recommendations for restaurants and hotels.

Leo is still under development, but it has learned to perform many kinds of tasks, including:

* Providing Information about Tourist Attractions
* Recommending popular tourist spots based on user preferences.
* Giving descriptions and details about attractions, such as historical significance, opening hours, admission fees, etc.
* Offering Directions and Navigation
* Providing directions to reach specific destinations or attractions.
* Offering alternative transportation options like public transit, walking, or taxi.
* Assisting with Accommodation
* Recommending hotels, hostels, or vacation rentals based on user preferences (e.g., budget, location, amenities).
* Suggesting Activities and Events
* Recommending local events, festivals, or activities happening during the user's visit.
* Answering Frequently Asked Questions (FAQs)
* Responding to common queries like currency exchange rates, local customs, emergency contacts, etc.

Here are some examples of how you can use Leo :

**Website Integration**: Embed Leo on your tourism website to provide real-time assistance to visitors. Leo can help with recommendations, directions, and information about local attractions.

**Mobile App Integration**: Include Leo in a mobile app for tourists. This allows users to get on-the-go recommendations, find nearby points of interest, and get assistance with travel-related queries.

**Hotel Concierge Service**: Integrate Leo into a hotel's website or app to serve as a digital concierge. Leo can help guests with information about the hotel, local attractions, dining options, and more.

**City Tourism Board Website**: Incorporate Leo into the website of a city's tourism board to provide personalized recommendations for visitors. Leo can suggest popular attractions, restaurants, and events.

**Cruise Line Customer Service**: Implement Leo in a cruise line's customer service platform to provide information about cruise itineraries, on-board services, and shore excursions.

To use Leo, simply go to the website and start chatting. You can also access Leo through the IBM Cloud Chatbot Watson Assistant app.

|  |  |
| --- | --- |
|  |  |

\*\*\* Images are subjected to their respective owners. These are demo

***Performance Metrics***

|  |  |  |
| --- | --- | --- |
| ***S. No*** | ***Metrics*** | ***Description*** |
| PM1 | Accuracy | The accuracy of the Leo in the tourism domain refers to the correctness of information and responses provided to users. It reflects the overall correctness of the chatbot's answers compared to the accurate and expected information about travel, destinations, accommodations, and other relevant queries. Higher accuracy signifies a lower rate of incorrect or irrelevant responses. |
| PM2 | Precision | Precision in Leo context relates to the relevance and correctness of the provided information. It indicates the proportion of accurate responses to the total number of responses given. A high precision indicates that when the chatbot provides an answer, it's more likely to be relevant and accurate. |
| PM3 | Recall | Recall represents the Leo's ability to retrieve and provide relevant information from its knowledge base when responding to user queries. It shows how well the Leo can recall and offer appropriate information. High recall indicates that the chatbot can retrieve a significant amount of relevant data for a wide range of user inquiries. |
| PM4 | Response time | Response time in a Leo chatbot is the duration taken by the chatbot to provide an answer or assistance to a user's query. For a tourism chatbot, users expect relatively quick responses to their inquiries. A good response time ensures a seamless and efficient user experience, enhancing user satisfaction. |
| PM5 | User satisfaction | User satisfaction in a tourism-based chatbot context is a measure of how content and pleased users are with the Leo's performance. It encompasses the overall user experience, considering factors such as the accuracy of information, response time, the chatbot's ease of use, helpfulness, friendliness, and the ability to resolve queries effectively. High user satisfaction indicates that the chatbot meets or exceeds users' expectations, thereby enhancing their experience and trust in the service. |

Here is a simple example of how an IBM Watson Assistant Tourism-based chatbot might work:

User: Hi! I'm planning a trip to New York City. Can you recommend some popular attractions?

Chatbot: Hi! Sure, I'd be happy to help. In New York City, you should definitely check out the Statue of Liberty, Central Park, and the Empire State Building. Do you have any specific interests?

User: I love art and history.

Chatbot: Great! You might enjoy the Metropolitan Museum of Art and the Museum of Natural History. They have fantastic collections. Would you like directions to any of these places?

User: Yes, please. How do I get to the Statue of Liberty?

Chatbot: To get to the Statue of Liberty, you'll want to take a ferry from Battery Park in Lower Manhattan. The ferry departs regularly. Would you like more information?

User: Yes, when does the first ferry leave?

Chatbot: The first ferry to the Statue of Liberty leaves at 9:30 AM. It's recommended to arrive a bit earlier to allow time for boarding. Is there anything else you'd like to know?

User: No, that's all. Thanks for your help!

Chatbot: You're welcome! Have a great trip to New York City! If you have any other questions, feel free to ask. Safe travels!

This is just a simple example, but it illustrates how IBM Watson Assistant chatbots can be used to help users with a variety of Tourism-related tasks.

**ADVANTAGES AND DISADVANTAGES:**

***Advantages***

Chatbots have a number of advantages over traditional customer service channels, such as phone and email. These advantages include:

1. ***Availability***: Chatbots are available 24/7/365, which means that customers can get help whenever they need it.
2. ***Scalability:*** Chatbots can handle a large volume of inquiries simultaneously, which can help businesses to reduce wait times and improve customer satisfaction.
3. ***Personalization:*** Chatbots can be personalized to meet the needs of individual customers. For example, chatbots can remember a customer's past interactions and provide tailored recommendations.
4. ***Efficiency:*** Chatbots can automate many repetitive tasks, such as answering frequently asked questions and booking appointments. This can free up customer service agents to focus on more complex issues.
5. ***Cost savings:*** Chatbots can help businesses to save money on customer service costs. Chatbots are less expensive than hiring and training human customer service agents.

In addition to these advantages, chatbots also offer a number of benefits for customers. For example, chatbots can:

* ***Provide instant support***: Chatbots can provide customers with instant support, without the need to wait on hold or send an email.
* ***Be accessed from anywhere:*** Chatbots can be accessed from anywhere with an internet connection. This makes it easy for customers to get help when they are traveling or when they are away from their computer.
* ***Be used to resolve complex issues:*** Chatbots can be used to resolve even the most complex issues. Chatbots can access and process information from a variety of sources, such as knowledge bases and CRM systems. This allows chatbots to provide customers with the most accurate and up-to-date information.

Overall, chatbots offer a number of advantages for both businesses and customers. Chatbots can help businesses to improve customer service, reduce costs, and increase efficiency. Chatbots can also help customers to get the support they need quickly and easily.

***Disadvantages:***

Chatbots have several advantages, but they also have some disadvantages. Here are some of the disadvantages of chatbots:

1. ***Limited capabilities:*** Chatbots are still under development, and they are not yet able to handle all of the tasks that a human customer service agent can. For example, chatbots may not be able to resolve complex issues or to provide personalized support.
2. ***Lack of empathy:*** Chatbots can sometimes seem cold or impersonal, and they may not be able to understand the emotions of customers. This can lead to frustration and dissatisfaction on the part of customers.
3. ***Security risks:*** Chatbots can be vulnerable to hacking and other security threats. This could put customer data at risk.
4. ***Cost:*** Developing and maintaining chatbots can be expensive, especially for small businesses.

Despite these disadvantages, chatbots offer a number of advantages for both businesses and customers. Overall, chatbots are a valuable tool that can be used to improve customer service, reduce costs, and increase efficiency.

Here are some tips for overcoming the disadvantages of chatbots:

* Be clear about the capabilities of your chatbot. Let customers know what the chatbot can and cannot do. This will help to avoid frustration and disappointment.
* Use your chatbot to handle simple tasks. This will free up your human customer service agents to focus on more complex issues.
* Make sure your chatbot is secure. Use strong encryption and other security measures to protect customer data.
* Invest in high-quality chatbot development. This will help to ensure that your chatbot is reliable and easy to use.

# **CONCLUSION**

In conclusion, implementing a chatbot for tourism using cloud-based solutions like Watson Assistant offers a multitude of advantages for both businesses and travelers. These chatbots provide 24/7 support, cost-efficiency, scalability, and multilingual capabilities. They can deliver personalized recommendations, instant responses, and valuable data collection and analysis, ultimately enhancing the overall customer experience. Integration with other systems, such as booking engines, ensures a streamlined reservation process, while maintaining consistency and reducing language barriers.

The following are some of the benefits of using a chatbot for tourism using cloud Watson Assistant:

\* **24/7 availability**: Chatbots can be accessed 24/7, providing tourists with information and assistance whenever they need it.

\* **Multi-lingual support**: Chatbots can be configured to support multiple languages, making them accessible to tourists from all over the world.

\* **Personalization**: Chatbots can be personalized to meet the needs of individual tourists. For example, a chatbot could learn a tourist's interests and recommend attractions, restaurants, and activities accordingly.

\* **Scalability**: Chatbots can be scaled up or down to meet demand, making them ideal for businesses of all sizes.

\* **Cost-effectiveness**: Chatbots can be more cost-effective than traditional customer service channels, such as phone and email.

\* **Improved customer satisfaction**: Chatbots can improve customer satisfaction by providing tourists with quick and easy access to information and assistance.

\* **Increased revenue**: Chatbots can help businesses to increase revenue by upselling and cross-selling products and services.

Here are some specific examples of how a chatbot for tourism using cloud Watson Assistant can be used to benefit both tourists and businesses:

\* A chatbot can be used to provide tourists with information about attractions, restaurants, and activities in their destination city. The chatbot can also help tourists book reservations and purchase tickets.

\* A chatbot can be used to help tourists navigate their destination city by providing them with directions and transportation information.

\* A chatbot can be used to help tourists communicate with locals by translating between different languages.

\* A chatbot can be used to help businesses collect feedback from tourists and to improve their products and services.

\* A chatbot can be used to help businesses generate leads and sales by promoting their products and services to tourists.

Overall, a chatbot for tourism using cloud Watson Assistant can provide a number of benefits for both tourists and businesses. Chatbots can help tourists to get the most out of their trip and can help businesses to improve their customer service and increase their revenue.

**FUTURE SCOPE**

The future of chatbot development based on Tourism is very promising. There are a number of areas where chatbots can be used to improve the T experience, including:

**Customer Support**: Chatbots can provide instant assistance to travelers, helping them with inquiries, bookings, and recommendations.

**Language Translation**: Chatbots can bridge language barriers by offering real-time translation services for tourists in foreign countries.

**Local Recommendations**: They can suggest local attractions, restaurants, and events based on user preferences and location.

**Travel Planning**: Chatbots can help users plan their trips by providing information on itineraries, visa requirements, and travel restrictions.

**Booking Assistance**: They can assist with booking flights, hotels, and transportation, simplifying the reservation process.

Here are some specific examples of how chatbots can be used to improve the tourism experience:

**Virtual Tour Guides**: Chatbots can provide guided tours through text or voice, offering information about historical sites, landmarks, and local culture.

**Interactive Maps**: Chatbots can assist tourists in navigating through cities, suggesting the best routes and highlighting points of interest on interactive maps.

**Language Learning**: Chatbots can help travelers learn basic phrases in the local language, making communication with locals easier and more enjoyable.

**Weather Updates**: They can provide real-time weather forecasts and recommendations for suitable activities based on weather conditions.

**Personalized Recommendations**: Chatbots can offer tailored suggestions for restaurants, activities, and entertainment based on a user's preferences and past choices.

These examples demonstrate the versatility of chatbots in offering a more personalized and convenient tourism experience.

The continual advancement of chatbot for tourism using cloud Watson Assistant is being driven by a number of factors, including:

\* **Improvements in natural language processing (NLP)**: NLP is the ability of computers to understand and respond to human language. As NLP algorithms improve, chatbots are able to understand more complex and nuanced language, which allows them to provide more comprehensive and informative responses to tourists' queries.

\* **Advances in machine learning (ML)**: ML algorithms allow computers to learn from data without being explicitly programmed. Chatbots can use ML algorithms to learn from their interactions with tourists and improve their performance over time. For example, a chatbot can use ML to learn which attractions and restaurants are most popular with tourists and then recommend those attractions and restaurants to new tourists.

**\* The growth of the cloud computing market**: Cloud computing provides chatbots with access to scalable and reliable computing resources. This allows chatbots to handle large volumes of traffic and provide a consistent user experience for all tourists.

Here are some specific examples of how the continual advancement of chatbot for tourism using cloud Watson Assistant is leading to new and innovative features and capabilities:

\* **Multilingual support**: Chatbots are now able to support multiple languages, which makes them accessible to tourists from all over the world.

\* **Context-aware responses**: Chatbots are now able to understand the context of a conversation and provide more relevant and informative responses. For example, if a tourist is asking questions about a specific attraction, the chatbot can provide detailed information about that attraction, such as its location, hours of operation, and admission fees.

\* **Personalized recommendations**: Chatbots are now able to learn from a tourist's interests and preferences and provide personalized recommendations for attractions, restaurants, activities, and events.

\* **Integration with other systems**: Chatbots can now be integrated with other systems, such as CRM systems and travel booking systems. This allows chatbots to provide tourists with more seamless and personalized experiences.

Overall, the continual advancement of chatbot for tourism using cloud Watson Assistant is making them more powerful, intelligent, and useful than ever before. Chatbots are now able to provide tourists with a wide range of information and assistance, and they are able to do so in a personalized and efficient manner. As the technology continues to develop, we can expect to see chatbots play an even more important role in the tourism industry in the years to come.

**SOURCE CODE:**

import json

from ibm\_watson import AssistantV2

from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator

# Set up your Watson Assistant credentials apikey = "YOUR\_API\_KEY" url = "YOUR\_ASSISTANT\_URL"

assistant\_id = "YOUR\_ASSISTANT\_ID"

# Authenticate with Watson Assistant authenticator = IAMAuthenticator(apikey)

assistant = AssistantV2( version='2021-06-14',

authenticator=authenticator

)

assistant.set\_service\_url(url)

# Create a session

response = assistant.create\_session(assistant\_id=assistant\_id) session\_id = response.get\_result()['session\_id']

# Define a function to send a message to Watson Assistant def send\_message(message): response = assistant.message( assistant\_id=assistant\_id, session\_id=session\_id, input={

'message\_type': 'text',

'text': message

}

)

return response.get\_result()

# Interaction with the chatbot user\_input = "Find hotels in Paris"

response = send\_message(user\_input)

# Extract and display Watson Assistant's response

bot\_response = response['output']['generic'][0]['text'] print("Chatbot: " + bot\_response)

# Remember to end the session when you're done

assistant.delete\_session(assistant\_id=assistant\_id, session\_id=session\_id) from flask import Flask, render\_template

from watson\_developer\_cloud import AssistantV2

app = Flask(\_name\_)

assistant = AssistantV2( iam\_apikey='YOUR\_API\_KEY',

url='https://{region}.watsonconversation.com/assistant/api'

)

deployment\_id = assistant.deploy\_chatbot( assistant\_id='YOUR\_ASSISTANT\_ID'

)['deployment\_id']

@app.route('/chatbot') def chatbot():

embed\_code = assistant.generate\_chatbot\_embed\_code( deployment\_id=deployment\_id

)

return render\_template('chatbot.html', embed\_code=embed\_code)

if \_name\_ == '\_main\_': app.run(debug=True)

**APPENDIX:**

Source code @github:

<https://github.com/Harini-balasubramanian/CHATBOT_CLOUD/blob/main/watson_assistant>.

pycode