**LAB-**04 **Create an ‘Ask me anything’ agent for your employees**

**Empower employees with answers that matter – Build an intelligent agent that connects to your knowledge and data**

# Lab Details

|  |  |  |  |
| --- | --- | --- | --- |
| Level | Persona | Purpose | Estimated time to complete |
| 200 | Maker |  | 20 minutes |

## Summary of targets

In this lab, you'll create an “Ask me anything” agent designed to support employees with fast, accurate answers across multiple internal systems.

| Use case/topic | Sujet |
| --- | --- |
| Create an ‘Ask me Anything’ agent and add SharePoint as a knowledge source | |
| Configure suggested prompts | |
| Deploy to Microsoft 365 Copilot | |
| Learn to prioritize knowledge sources | |
| Add ServiceNow incidents | |
| Add custom knowledge | |
| AI summary and review of meeting notes | |
|  | |

# Use Case #1: Create an ‘Ask Me Anything’ agent

## Summary of tasks

In this section, you’ll learn how to access the Solutions area of Copilot Studio, create a new solution, new publisher, and set the solution as default.

**Scenario**: Properly setup your development environment so that you can later easily package and deploy your agents to other environments.

## Step-by-step instructions

1. Navigate to the Copilot Studio **home** **page**.

<https://aka.ms/MCSStart>

1. Go to the **Solutions** menu (located in the left-hand menu under the ellipsis **…**)
2. Select the **solution** you had created.
3. Select **New**, and choose **Agent**.
4. Select **Skip to configure**
5. Name your agent **Ask Me Anything**
6. **Create**
7. Go to **Knowledge**
8. Select **SharePoint**, and add this URL

|  |
| --- |
| https://copilotstudiotraining.sharepoint.com/Shared%20Documents/Forms/AllItems.aspx |

1. Choose **Add**
2. Test your agent with a **question**

|  |
| --- |
| What's the Northwind Health Plus Benefits plan? |

**Notice how relevant it is, and how the citations are accurate**

# Use Case #2: Configure suggested prompts

## Summary of tasks

Suggest things your agent can do to your end-users.

## Step-by-step instructions

1. Go to the **overview tab** for your agent
2. Add suggested prompts, such as:

**Ask me about benefits**

|  |
| --- |
| What's the Northwind Health Plus Benefits plan? |

**Ask me about policies**

|  |
| --- |
| What's the leave policy in France? |

**Help fill out meeting notes**

|  |
| --- |
| Capture and structure meeting notes |

**Check on an incident**

|  |
| --- |
| What’s the status of INC0007001? |

1. **Save** the prompts

# Use Case #3: Deploy to Microsoft 365 Copilot

## Summary of tasks

Make the agent available in Microsoft 365 Copilot for your users.

## Step-by-step instructions

1. Go to the **agent**
2. Select **Publish**
3. Go to **Channels**
4. Select **Teams and Microsoft 365 Copilot**
5. Select **Add Channel**
6. Select **See agent in Microsoft 365**
7. Give it a few tries / minutes the first time you deploy to Teams so it deploys correctly. Once ready, select **Add**.
8. Test the agent with the **benefits** prompt

A screenshot of a computer

AI-generated content may be incorrect.

# Use Case #4: Knowledge prioritization techniques

## Summary of tasks

In this section, you’ll learn different techniques to help prioritize the right knowledge sources based on context.

**Why do you need this? Because the SharePoint site has different folder for different country leave policies (see** [**https://copilotstudiotraining.sharepoint.com/Shared Documents/Leave policies/**](https://copilotstudiotraining.sharepoint.com/Shared%20Documents/Leave%20policies/)**). So asking a question like:**

|  |
| --- |
| What's the leave policy? |

**Doesn’t return relevant results and seem to randomly pick a leave policy for a country.**

## Step-by-step instructions

Let’s start by asking for the user country at the beginning of the conversation. In many situations, this can be obtained from context, either passed from a web page to the agent, or by making an initial call (for example to the Entra ID connector) to get more information about the logged-in user and their location.

1. In your ‘**Ask Me Anything**’ agent, go to **Topics**, and **System topics**
2. Open the **Conversation Start** topic
3. Right after the **trigger**, add a **question** node

|  |
| --- |
| What’s is your country |

1. Configure the **France**, **Germany**, **India**, **USA** as options for the user.
2. Select the **Var1** variable, rename it to **Country**, and make it **Global**.
3. Create a new **topic**, and name it **Leave policy**
4. Provide the below description for the trigger

|  |
| --- |
| Use this tool for questions about leave policy and time off |

1. Add a new **node**. Select **Advanced** > **Generative answer**.
2. For input, select the **Activity.Text** system variable.
3. Go to the **properties** of the **Create** **generative answers node**.
4. Select **Search only selected sources** (there should be none selected)
5. Under **classic data**, for **SharePoint**, toggle **Manual input** to **Formula**
6. Select the **…** and go to the **formula** tab.
7. **Use the below formula to make the SharePoint URL dynamic based on the country selected by the user. Select Insert once done.**

|  |
| --- |
| [ "https://copilotstudiotraining.sharepoint.com/Shared Documents/Leave policies/" & Global.Country ] |

1. **Save** your topic
2. **Refresh** the test pane, select any **country**, and ask

|  |
| --- |
| What’s the leave policy? |

**Try multiple times and validate that the results are correct for the different countries.**

💡 **PRO TIPS:**

* There are multiple ways to do knowledge prioritization. You could add to your agent instructions your Country global variable, and instruct the agent to “Always ground your HR questions knowledge search with the user location: {Global.Country}”
* If formulas are too much code, you could also use condition nodes and configure different branches based on the selected country.

# Use Case #5: Add ServiceNow incidents

## Summary of tasks

In this section, you’ll configure the connection to ServiceNow to retrieve incident details.

## Step-by-step instructions

1. Go to the **agent**
2. Go to **Tools**, and **Add tool**
3. Select **ServiceNow** and choose **List records**
4. Select **Add to agent**
5. Open **List records**.
6. Under **Additional details**, change **Authentication** to **Agent author authentication**.
7. Rename to **Get ServiceNow ticket details**
8. Change description to **Gets the details of an incident using its incident number**
9. For **Record Type**, set a **Custom value** and choose **Incident**
10. Add input: **Query**Select **Customize** and use this **Description**

|  |
| --- |
| The output of this variable is the concatenation of numberCONTAINS and the incident number.  E.g., "numberCONTAINSINC0007001".  Only the incident number should be prompted and obtained from the user (e.g., INC0007001) |

1. Add input: **Limit**Select **Custom Value** and set **1**
2. Choose **Add**
3. Test your agent with a **question**

|  |
| --- |
| what's the status of case INC0000059 |

**Notice how the agent automatically formats the user response in a user-friendly way. accurate**

# Use Case #6 Add custom knowledge

## Summary of tasks

In this section, you’ll configure any third-party knowledge to enrich knowledge results.

## Step-by-step instructions

1. Go to the **agent**
2. Go to **Topics,** add a **new topic**
3. Call your topic **Custom Knowledge**
4. Go to **More …** and open **Code editor**

**Replace with the below YAML code**

|  |
| --- |
| kind: AdaptiveDialog  beginDialog:    kind: OnKnowledgeRequested    id: main  inputType: {}  outputType: {} |

1. Add a new **node** > **Advanced**, **Send HTTP request**
2. For **URL**, choose the **CustomKnoweldgeEndpoint** environment variable
3. Method: **Post**
4. Response data type: **Tabe**
5. **Edit Schema**

|  |
| --- |
| kind: Table  properties:  @odata.type: String  Article: String  Id: String  Title: String  URL: String |

1. **Edit Headers and body**
2. Body: **JSON content**
3. Edit **formula**

|  |
| --- |
| {  SearchQuery: System.Activity.Text  } |

1. Save response as **KnowledgeResults** variable
2. Add a **Set a variable value** node
3. Select System > **SearchResults**
4. To value **formula**:

|  |
| --- |
| ForAll(      Topic.KnowledgeResults,      {          Title: Title,          Content: Article,          ContentLocation: URL      }  ) |

1. **Save**
2. **Test**

|  |
| --- |
| How do I update the vendor information for an accounts payable invoice? |

# Use Case #7 Upload meeting notes for a human-reviewed AI summary

## Summary of tasks

In this section, you’ll learn how to upload files and run files through a prompt for analysis, and how to user adaptive card to ask for a human review for edits before submitting a meeting report.

## Step-by-step instructions

1. Go to the **agent**
2. Go to **Topics,** add a **new topic**
3. Call your topic **Meeting Notes**
4. Describe what the topic does: **Help with meeting notes**
5. Add a **Question** node

|  |
| --- |
| Please upload your meeting notes |

Identify: **File**

Rename variable as **File**

1. Add a **new node > Add a tool > New prompt**
2. Name: **Meeting AI Notes**
3. Add Content: **Image or Document**

As a sample, use **Meeting Minutes.pdf** from [aka.ms/MCSWorkshopLabAssets](https://aka.ms/MCSWorkshopLabAssets)

1. In in the **instructions**, add

|  |
| --- |
| Using the content in the document , create an output JSON file with these different string properties (each of them are a single text string)  - title  - description of the meeting  - date and time  - attendees  - actions |

1. Output: **JSON**
2. **Test** and **Save** the prompt
3. Name the output variable as **MeetingAINotes**
4. Add a new node, **Ask with Adaptive Card**
5. Toggle format to **Formula**, and paste:

|  |
| --- |
| {  type: "AdaptiveCard",  version: "1.5",  '$schema': "https://adaptivecards.io/schemas/adaptive-card.json",  body: [  {  type: "TextBlock",  text: "Please review and confirm the meeting minutes",  weight: "Bolder",  size: "Medium"  },  {  type: "Input.Text",  id: "meetingTitle",  label: "Meeting title",  value: Topic.MeetingAINotes.structuredOutput.title,  outputType: "Text"  },  {  type: "Input.Text",  id: "attendees",  label: "Attendees",  value: Topic.MeetingAINotes.structuredOutput.attendees,  outputType: "Text"  },  {  type: "Input.Text",  id: "date",  label: "Date",  value: Topic.MeetingAINotes.structuredOutput.date\_0024b37cd5ce9ac5b83f58301743156a40f,  outputType: "Text"  },  {  type: "Input.Text",  id: "summary",  label: "Meeting summary",  isMultiline: true,  value: Topic.MeetingAINotes.structuredOutput.descript80c0f9ddffa9e7f43d05a2f022d3aa11,  outputType: "Text"  },  {  type: "Input.Text",  id: "nextSteps",  label: "Next steps",  isMultiline: true,  value: Topic.MeetingAINotes.structuredOutput.actions,  outputType: "Text"  }  ],  actions: [  {  type: "Action.Submit",  title: "Confirm minutes"  }  ]  } |

1. **Save**
2. **Test** your agent

|  |
| --- |
| Please upload your meeting notes |