Contents

[OVERVIEW 3](#_Toc216026821)

[CREATING COPILOT AGENT 3](#_Toc216026822)

[CREATING AGENT USING COPILOT METHOD 3](#_Toc216026823)

[CREATING AGENT USING CONFIGURE METHOD 3](#_Toc216026824)

[SETTING IN THE AGENTS 4](#_Toc216026825)

[CONFIGURING KNOWLEDGE SOURCE 5](#_Toc216026826)

[TESTING AGENT WITH KNOWLEDGE SOURCE 6](#_Toc216026827)

[TOPICS 6](#_Toc216026828)

[TYPES OF TOPICS 6](#_Toc216026829)

[SYSTEM TOPICS 7](#_Toc216026830)

[CUSTOM TOPICS 8](#_Toc216026831)

[NODES 11](#_Toc216026832)

[TOOLS 11](#_Toc216026833)

[SUPPORTED TOOLS 12](#_Toc216026834)

[PROMPT TOOL 12](#_Toc216026835)

[REST API 12](#_Toc216026836)

[AGENT FLOW 14](#_Toc216026837)

[AGENT VERSUS AGENT FLOW 14](#_Toc216026838)

[WHAT IS AN AGENT IN COPILOT STUDIO? 14](#_Toc216026839)

[WHAT IS AN AGENT FLOW? 14](#_Toc216026840)

[REAL WORLD AGENTS 15](#_Toc216026841)

[STUDY SCHEDULER AGENT 15](#_Toc216026842)

[CREATE THE AGENT – AGENT CONFIGURATION 15](#_Toc216026843)

[CREATE TOPICS & ADD PROMPTS 16](#_Toc216026844)

[STUDY SCHEDULER & DB AGENT 25](#_Toc216026845)

[PROVISION SQL DB IN AZURE 26](#_Toc216026846)

[ADDING ACTIONS TO THE TOPIC 30](#_Toc216026847)

[TEST AGENT 33](#_Toc216026848)

[DOCUMENT INSIGHT AGENT 34](#_Toc216026849)

[EMAIL NOTIFIER AGENT – USING AGENT FLOW 35](#_Toc216026850)

[CREATE AGENT FLOW 36](#_Toc216026851)

[INVOICE SCANNER & DATA ENTRY 41](#_Toc216026852)

[CREATE A DATAVSERSE TABLE 42](#_Toc216026853)

[CREATE AGENT 42](#_Toc216026854)

[ADD TOPIC 43](#_Toc216026855)

[MOVIE MATCH AGENT 45](#_Toc216026856)

[HR ASSISTANT USING DATAVERSE MCP 45](#_Toc216026857)

[MCP OVERVIEW 45](#_Toc216026858)

[AGENT OVERVIEW 46](#_Toc216026859)

[CREATE TABLE IN DATAVERSE 49](#_Toc216026860)

[CREATING AGENT 51](#_Toc216026861)

[ADD DATAVERSE MCP SERVER AS A TOOL 52](#_Toc216026862)

[TEST AGENT 54](#_Toc216026863)

[INOVICE SCANNER AND DATA ENTRY AGENT 54](#_Toc216026864)

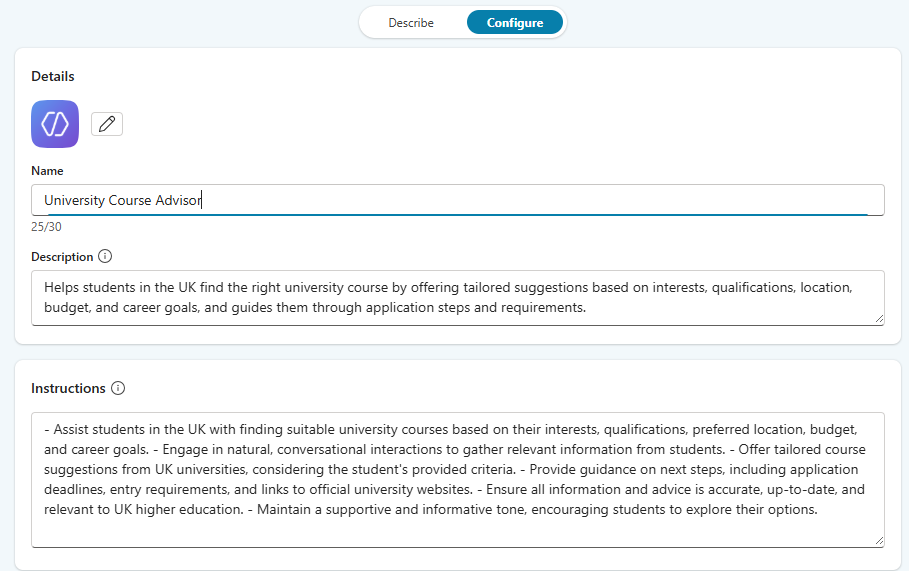
# OVERVIEW

* **Copilot Studio** is Microsoft’s platform for building and customizing **AI-powered copilots and agents**.

# CREATING COPILOT AGENT

## CREATING AGENT USING COPILOT METHOD

## CREATING AGENT USING CONFIGURE METHOD



|  |
| --- |
| **Name of Agent**: University Course Advisor  **Description**:  Helps students in the UK find the right university course by offering tailored suggestions based on interests, qualifications, location, budget, and career goals, and guides them through application steps and requirements.  **Instructions**:  - Assist students in the UK with finding suitable university courses based on their interests, qualifications, preferred location, budget, and career goals. - Engage in natural, conversational interactions to gather relevant information from students. - Offer tailored course suggestions from UK universities, considering the student's provided criteria. - Provide guidance on next steps, including application deadlines, entry requirements, and links to official university websites. - Ensure all information and advice is accurate, up-to-date, and relevant to UK higher education. - Maintain a supportive and informative tone, encouraging students to explore their options. |

A screenshot of a computer

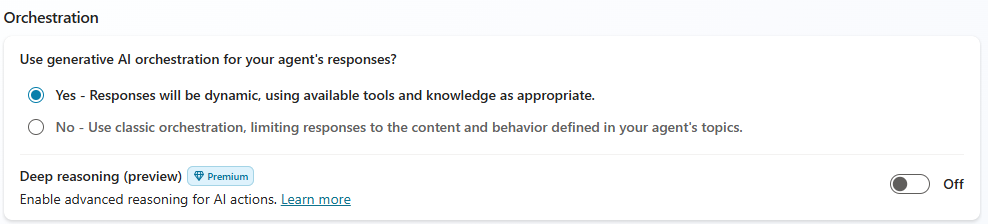
AI-generated content may be incorrect.

### SETTING IN THE AGENTS

A screenshot of a search engine

AI-generated content may be incorrect.

* The recommended way to is to use Generative AI Orchestration



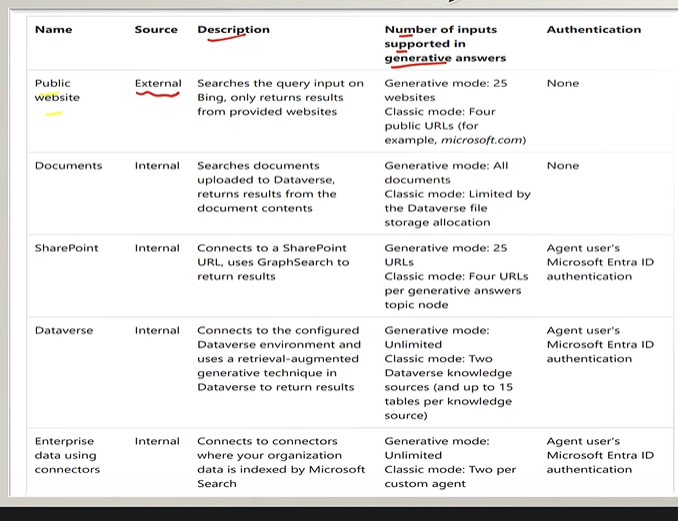
|  |
| --- |
| **What Generative AI Orchestration Means**  In **Microsoft Copilot Studio**, orchestration determines how an agent decides what to do when a user asks a question or triggers an event. There are two modes:   * **Classic Orchestration**: The agent matches the user’s query to predefined *topics* using trigger phrases and executes a fixed flow. It’s predictable but rigid. * **Generative AI Orchestration**: This is the advanced mode where the agent uses **AI-powered planning** to dynamically decide:   + Which **topics**, **tools**, and **knowledge sources** to use.   + Whether to involve **child or connected agents**.   + How to **generate responses** without relying on hardcoded messages.   Instead of following a static script, the agent leverages a **Large Language Model (LLM)** to:   * Understand user intent. * Combine multiple resources (topics, tools, knowledge). * Generate missing questions automatically to fill required inputs. * Produce a natural, context-aware response using all available data. [[learn.microsoft.com]](https://learn.microsoft.com/en-us/microsoft-copilot-studio/advanced-generative-actions)   **Bottom Line**: Generative AI orchestration makes Copilot Studio agents **intelligent and adaptive**, moving beyond rigid flows to dynamic, AI-driven conversations. |

### CONFIGURING KNOWLEDGE SOURCE

|  |
| --- |
| What is Grounding?   * Grounding is the process of **connecting AI model outputs to real-world, trusted data sources(like enterprise data, documents etc..)** so that responses are accurate, relevant, and context-aware. * Large Language Models (LLMs) like GPT are trained on general knowledge, but they don’t inherently know your organization’s private or up-to-date data. * Grounding solves this by injecting **specific, verified information** into the model’s reasoning |

Supported Knowledge Sources

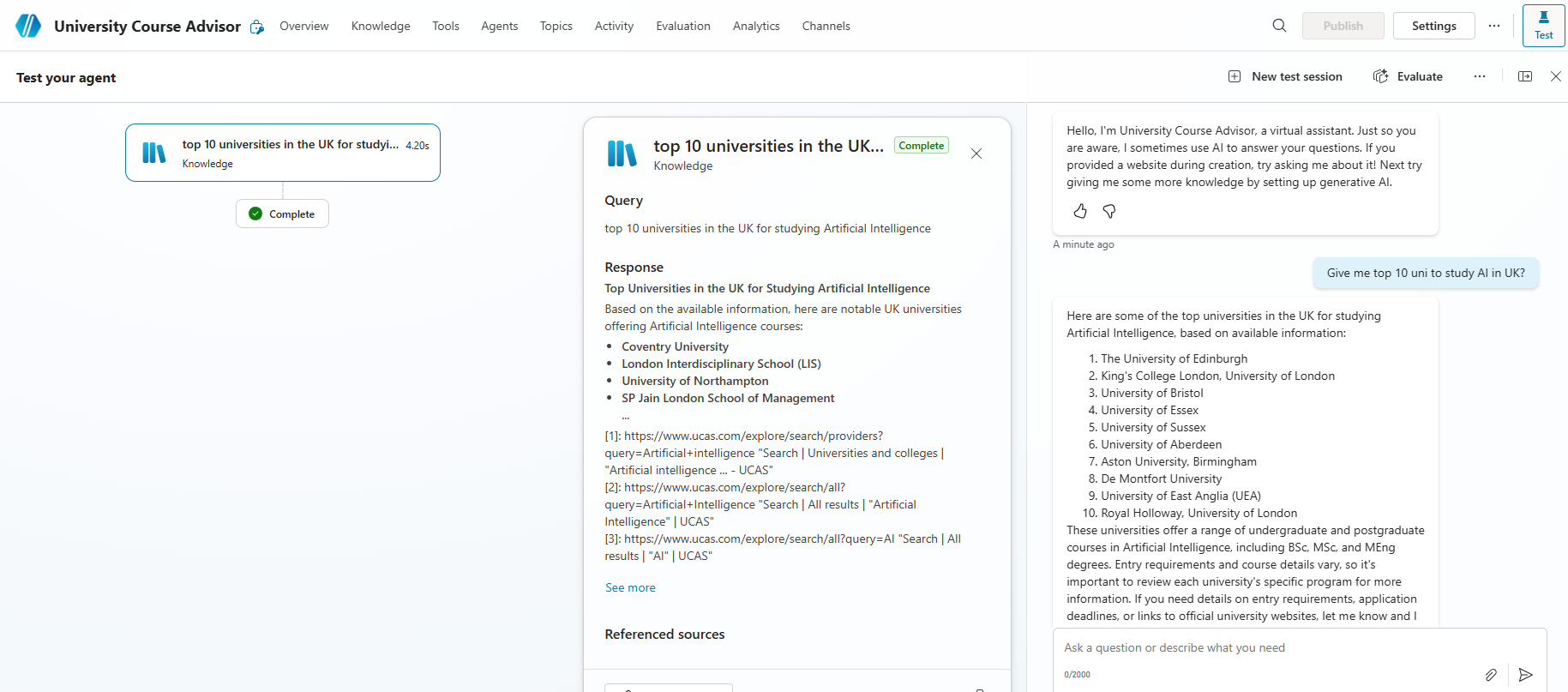
* **Public website**
* **Documents**
* **Sharepoint**
* **Dataverse** 
  + It’s a secure cloud data Platform by Microsoft
  + It organizes the data in table format
* **Enterprise Connectors**



**Note : Authentication is required for internal knowledge sources**

|  |
| --- |
| * If we want Agent to use a specific knowledge source. The need to disable the following option from Agent Settings * When we disable it – the agent will nor use the knowledge on which trend on neither the Bing web search as knowledge source |

### TESTING AGENT WITH KNOWLEDGE SOURCE



# TOPICS

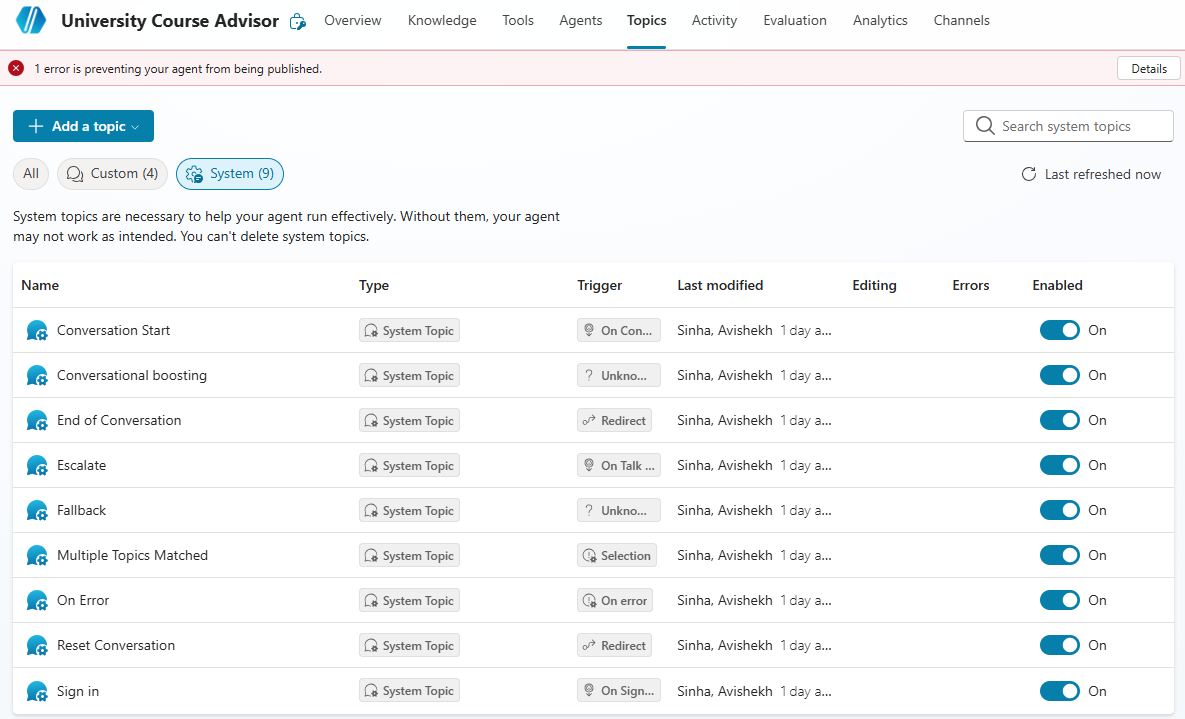
* Topics are **the core building blocks of a Copilot agent** in Microsoft Copilot Studio. They define **how a conversation flows** between the user and the AI agent. Think of topics as **conversation paths or competencies** that allow the agent to handle specific intents or scenarios.
* Each topic is triggered by recognizing a user’s intent through natural language understanding, directing conversation appropriately.

## TYPES OF TOPICS

There are 2 types of topics in MS Copilot

* **System Topics**: Predefined by Microsoft (e.g., greetings, end of conversation, escalation). *Cannot be deleted but can be turned off.*
* **Custom Topics**: Created by you to handle specific business scenarios. We *can add, edit, or delete these freely.*
* **Default Topics**: Basic conversational abilities like greeting, saying goodbye, or starting over.

### SYSTEM TOPICS



* System topics cannot be removed/deleted but can be tweaked

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a chat

AI-generated content may be incorrect.

### CUSTOM TOPICS

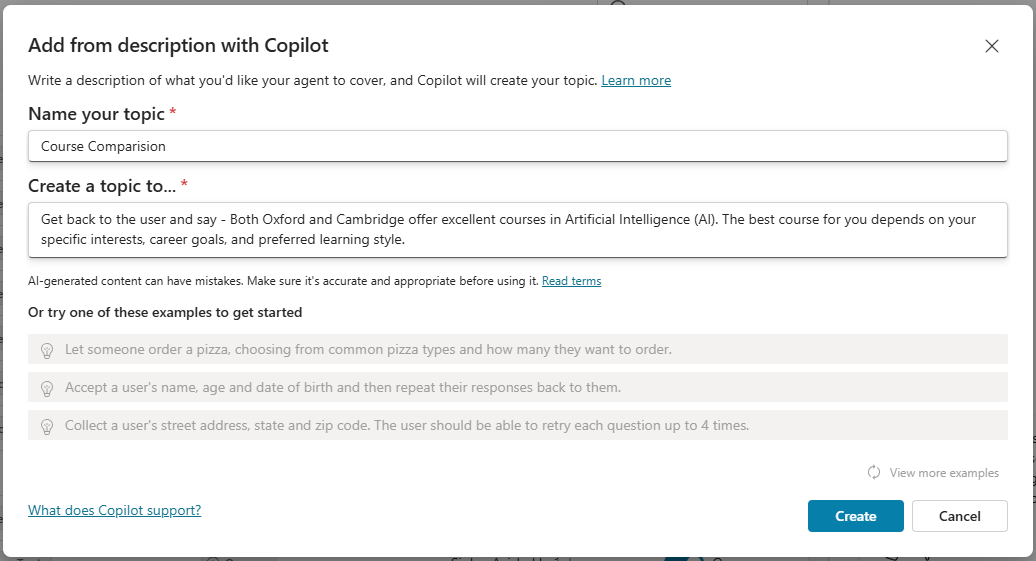
#### CREATING CUSTOM TOPICS

A screenshot of a computer

AI-generated content may be incorrect.

* Topics can be created either using natural language or manually configuring it.

##### CREATE TOPIC – ADD FROM DESCRIPTION WITH COPILOT



|  |
| --- |
| Example: Get back to the user and say - Both Oxford and Cambridge offer excellent courses in Artificial Intelligence (AI). The best course for you depends on your specific interests, career goals, and preferred learning style.  Then give multiple choice question to the user and ask the comparison needs to be on basis of Duration or Fee . If user says Duration then print Duration and structure are **nearly identical** between Oxford and Cambridge — the main differences lie in subject offerings, college systems, and style of supervision.  If user says Fee then print  · **Cambridge** charges an **additional college fee** for international students (~£11,000/year), while at Oxford this is usually included in the main tuition or presented differently.  · **Medicine at Cambridge** is **significantly more expensive** due to added clinical costs. |

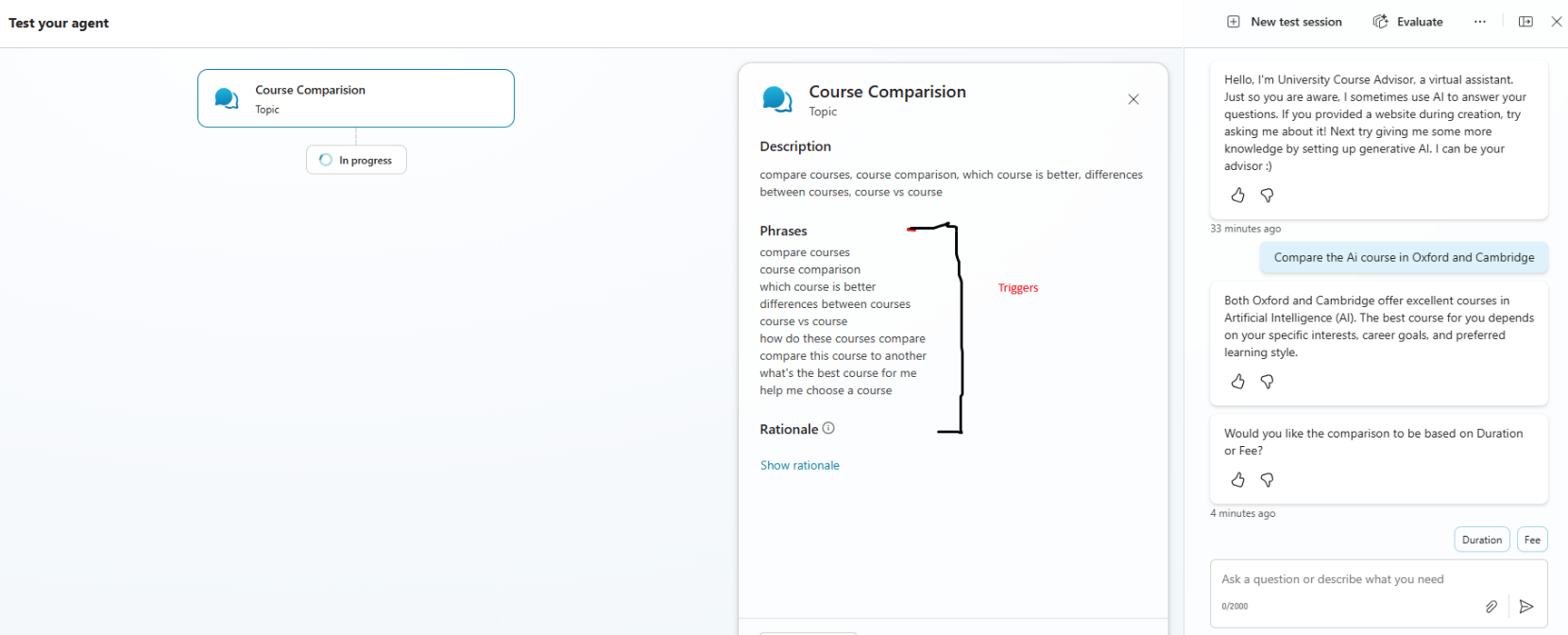
* This will create a workflow based on the above instructions

A screenshot of a computer

AI-generated content may be incorrect.

TESTING AGENT

* The Agent will get triggered based on the Trigger phrases. Note – these triggers phrases are auto created when we create agent with description.

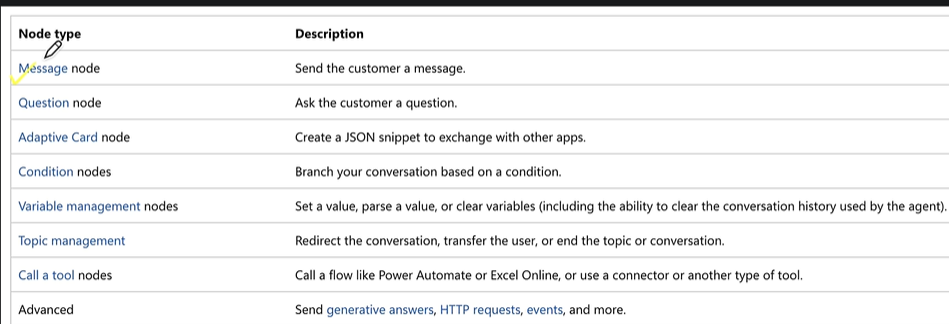


##### CREATE TOPIC – FROM BLANK

# NODES

* Nodes are the individual building blocks within topics that represent specific steps in the conversation.
* They control what happens at each point in the dialogue.

Supported Node Types



# TOOLS

* Tools are **the building blocks that enables the agent to interact with external system**
* Think of tools as **skills or actions** that agent can perform, like sending an email, checking weather, or calling an API.

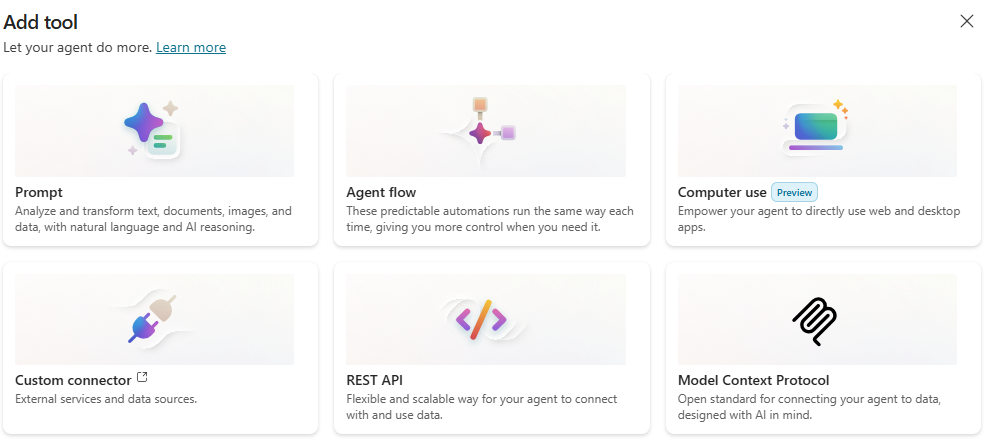
Why Use Tools?

* Topics handle conversation flow.
* Tools let your agent **connect to external systems** or **perform tasks** automatically.

Examples

* Send emails using the office 365 outlook connector
* Check current weather conditions and forecasts
* Read & write data from Dataverse tables(We can use MCP here)
* Read and Post messages to Teams

## SUPPORTED TOOLS



### PROMPT TOOL

### REST API



#### ADD CUSTOM CONNECTOR USING RESP API

A screenshot of a computer

AI-generated content may be incorrect.

|  |
| --- |
| * An **OpenAPI file** is a document that describes the structure and behavior of an API in a standardized way. * It follows the **OpenAPI Specification (OAS)**, which is widely used for RESTful APIs.   What is it?  It’s usually written in **JSON** or **YAML** format.  It defines:   * **Endpoints** (URLs your API exposes) * **Methods** (GET, POST, PUT, DELETE) * **Request and response formats** * **Authentication details** * **Data models** (schemas for input/output)   **Why is it important?**  **Standardization:** Makes APIs easy to understand and integrate.  **Automation:** Tools can generate documentation, SDKs, and tests from it.  **Interoperability:** Works with platforms like Copilot Studio, Postman, Swagger, etc.  **Example (YAML format)**  openapi: 3.0.0  info:    title: Sample API    version: 1.0.0  paths:    /users:      get:        summary: Get all users        responses:          '200':            description: Successful response            content:              application/json:                schema:                  type: array                  items:                    type: object                    properties:                      id:                        type: integer                      name:                        type: string  How does it relate to Copilot Studio?   * In **Copilot Studio**, you can use an OpenAPI file to **connect external APIs** to your Copilot. This allows Copilot to call those APIs and provide dynamic responses based on real data. |

## AGENT FLOW

### AGENT VERSUS AGENT FLOW

### WHAT IS AN AGENT IN COPILOT STUDIO?

An **Agent** is essentially a conversational AI bot built in Copilot Studio. It:

* Handles **natural language interactions** with users.
* Can **reason over data**, answer questions, and perform tasks.
* Is designed for **enterprise-grade scenarios**, integrating with Microsoft 365, Teams, Power Platform, and external systems.
* Supports **multi-turn conversations**, orchestration of skills, and secure lifecycle management. [[CoPilot Studio Agent | SharePoint]](https://uhgazure.sharepoint.com/teams/UHCTEIAIML/SitePages/CoPilot-Studio-Agent.aspx?web=1)

Think of an **Agent** as the “brain” that interacts with users and decides what needs to be done.

### WHAT IS AN AGENT FLOW?

An **Agent Flow** is a **deterministic workflow** inside Copilot Studio that automates tasks. It:

* Consists of **triggers** (events that start the flow) and **actions** (tasks performed).
* Can be created using **natural language** or a **visual designer**.
* Executes **rule-based steps** for predictable outcomes (e.g., send Teams notification when an email arrives).
* Can be triggered manually, by schedule, or by other agents/events.
* Provides **end-to-end visibility** for monitoring and optimization.
* Uses **Copilot Studio capacity** for billing (not Power Automate license). [[learn.microsoft.com]](https://learn.microsoft.com/en-us/microsoft-copilot-studio/flows-overview), [[linkedin.com]](https://www.linkedin.com/pulse/mastering-agent-flows-copilot-studio-step-by-step-guide-dhruvin-shah-mtzxf/)

Think of an **Agent Flow** as the “hands” that carry out structured tasks when the Agent decides to act.

**🔍 Key Differences: Agent vs Agent Flow**

| **Aspect** | **Agent** | **Agent Flow** |
| --- | --- | --- |
| **Purpose** | Conversational AI bot for user interaction | Workflow automation for repetitive tasks |
| **Focus** | Understanding and responding to user queries | Executing deterministic actions based on triggers |
| **Creation** | Built in Copilot Studio with topics, prompts | Built in Copilot Studio using designer or natural language |
| **Behavior** | Dynamic, AI-driven reasoning | Rule-based, predictable execution |
| **Integration** | Can call multiple flows and orchestrate skills | Can be added as a tool inside an Agent |
| **Licensing** | Copilot Studio environment | Uses Copilot Studio capacity (no Power Automate license) |

In short:

* **Agent = Conversational intelligence + orchestration**
* **Agent Flow = Automation logic for tasks**

**✅ Practical Example:**

* **Agent**: A “Helpdesk Copilot” that chats with employees.
* **Agent Flow**: A workflow that updates a ticket in ServiceNow and sends an email when the agent detects an issue.

# REAL WORLD AGENTS

## STUDY SCHEDULER AGENT



### CREATE THE AGENT – AGENT CONFIGURATION

|  |  |
| --- | --- |
| **Agent Name** | **Study Scheduler Agent** |
| **Description** | *Helps students create a personalized study schedule by collecting subjects, presenting them as*  *a list, and gathering scheduling details such as start date, end date, number of study hours,*  *and start time. Generates a table of study schedules for the student.* |
| **Instructions** | ***Instructions***  *- Collect subjects to study from the user and convert them into a list.*  *- Present the subjects list to the user for confirmation or editing.*  *- Use an adaptive card to gather scheduling details: start date, end date, number of hours of study, and start time.*  *- Generate a table of study schedules based on the provided information, distributing study hours across the selected dates and subjects.*  *- Ensure the schedule is clear, organized, and easy for the student to follow.*  *- Respond in a friendly and supportive manner, guiding the user through each step.*  *- Do not repeat information already provided by the user.*  *- Only communicate in English.*  *- Do not answer questions unrelated to study scheduling or agent modification.* |

**NOTE: Disable the Web Search and generative Ai knowledge**

A screenshot of a computer

AI-generated content may be incorrect.

### CREATE TOPICS & ADD PROMPTS

Step 1: Add Triggers

|  |  |
| --- | --- |
| **What are triggers?**  **A trigger is what starts an action or flow in Copilot Studio. It tells the agent when to act.** | |
| * We need to provide some triggering factor(keywords), which will kick in topic | create new study schedule  new study schedule  create study plan  make a study timetable  generate a study routine  I want a new schedule |

Step 2: Adding Question

|  |  |
| --- | --- |
| **Question** | What subjects you want to study |
| **Identity** | User’s entire response |
| **Save user response as** | “**subjects**” : This is the variable where entire user response will be stored |



A screenshot of a computer

AI-generated content may be incorrect.

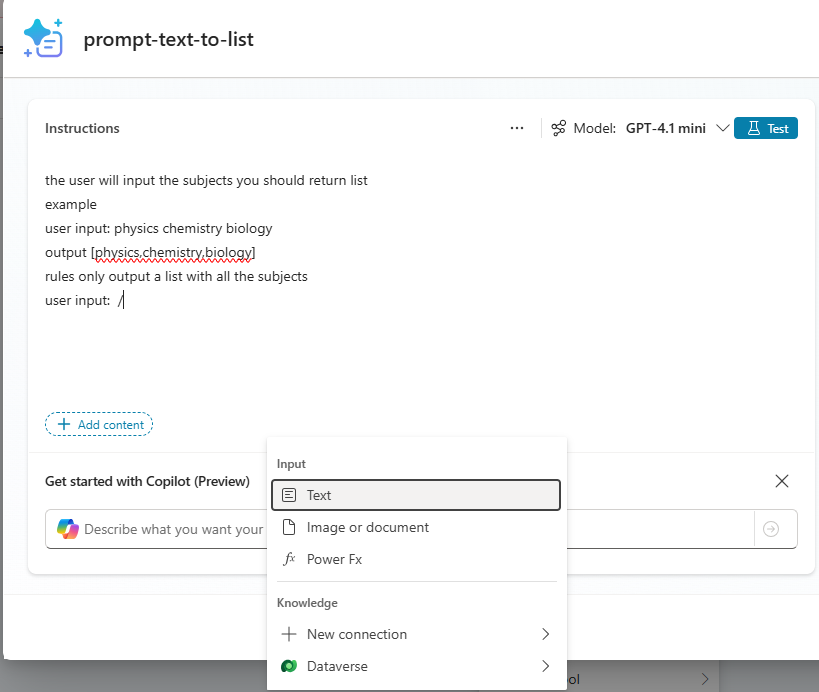
Step 3: ADD A Tool Prompt

* The prompt tool will convert the user’s response to a list.
* The reason for this is that turning the input into a structured format (like a list written as a string: ["Math", Science","History"]) makes it **much easier for the AI agent to process later**.
* Instead of handling raw, unstructured text, the agent can clearly see that the response is a **collection of items**.

A screenshot of a computer

AI-generated content may be incorrect.

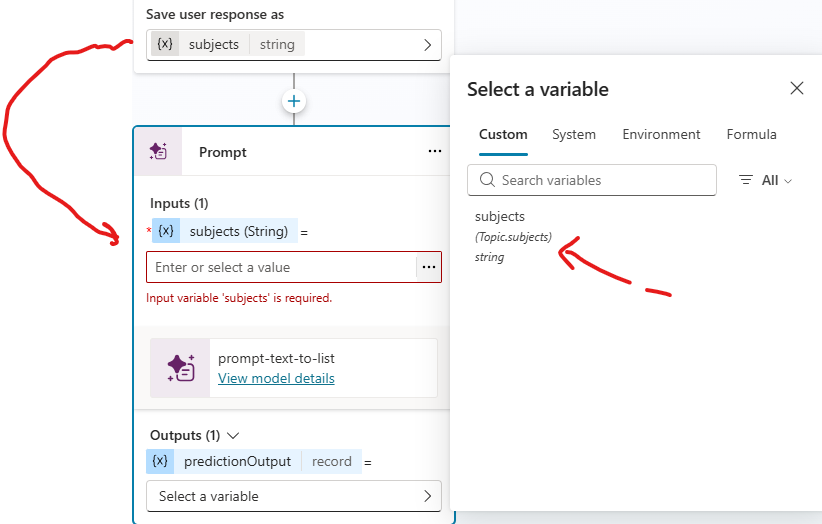
* The input to a prompt is the “subjects” variable received from the previous step.
* The value of the “subjects” variable is substituted in the prompt of prompt tool



A screenshot of a computer

AI-generated content may be incorrect.

|  |
| --- |
| TESTING PROMPT In prompt Tool  **The prompt tool will extract the subject names as list of subject form user’s response** |



* The list output from the prompt tool will be finally stored in the “**list\_of\_subjects**” variable

A screenshot of a computer

AI-generated content may be incorrect.

Step 4: ADD Adaptive Card

Once we take the subjects input – Now to create the study schedule need to take following inputs as **Adaptive Card**

The input in adaptive cards will be

* ***Start Date***
* ***End date***
* ***Time(in hours)***
* ***Start Time***

Now Add and edit the adaptive card

*A screenshot of a computer

AI-generated content may be incorrect.*

*A screenshot of a computer

AI-generated content may be incorrect.*

*A screenshot of a computer

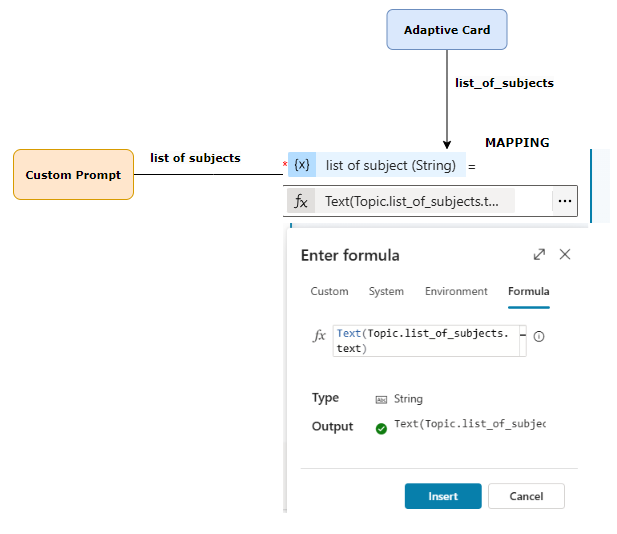
AI-generated content may be incorrect.*

|  |
| --- |
| JSON  **{**  **"type": "AdaptiveCard",**  **"$schema": "https://adaptivecards.io/schemas/adaptive-card.json",**  **"version": "1.5",**  **"body": [**  **{**  **"type": "Input.Date",**  **"label": "Start Date",**  **"id": "startDate"**  **},**  **{**  **"type": "Input.Date",**  **"label": "End Date",**  **"id": "endDate"**  **},**  **{**  **"type": "Input.Text",**  **"label": "Time in Hours",**  **"placeholder": "Enter hours of study",**  **"id": "time\_in\_hours"**  **},**  **{**  **"type": "Input.Time",**  **"label": "Start Time",**  **"id": "startTime"**  **}**  **],**  **"actions": [**  **{**  **"type": "Action.Submit",**  **"title": "Submit Schedule",**  **"data": {**  **"action": "submitSchedule"**  **},**  **"style": "positive"**  **}**  **]**  **}** |

Step 5: Add Custom PROMPT TO Convert Json To Table

* We receive JSON output from the Adaptive card. The custom prompt will convert the JSON into a tabular format.

|  |
| --- |
| **Custom Prompt**  You are given study schedule details in JSON format along with a start date and an end date.  Generate a full study schedule covering all days from start to end, and output the result in a  tabular format with the following columns:  (StudyDate, StartTime, EndTime, Subject, Notes, Status)  Rules:  Parse the time field for start time and duration (e.g., "9 pm 2 hrs" → StartTime = 21:00,  EndTime = 23:00).  Use the provided start\_date and end\_date to expand the schedule across multiple days  (inclusive).  For each day between start\_date and end\_date, create one row per subject. Randomize the  subjects i can study within the total hours of study on a day.  Notes = "Study **{subject}**".  Status = "Pending".  Output only as a neatly formatted table, no SQL queries.  Example Input JSON:  {  "start\_date": "2025-09-09",  "end\_date": "2025-09-12",  "time": "9 pm 2 hrs",  "subjects": ["physics", "chemistry", "maths"]  }  Here is the input JSON:  {  "start\_date":  "end\_date":  "time": ,  "subjects":  } |
|  |
| * In custom prompt use input variable in the prompt.(as shown above) * These input variables are assigned with a value which we are getting from the output of Adaptive card * Note the assignment is done by converting the values from adaptive card to string format, as prompt are expected to be in String format. |



A screenshot of a computer

AI-generated content may be incorrect.

* The final tabular output from custom prompt is stored in a variable “final\_message”.

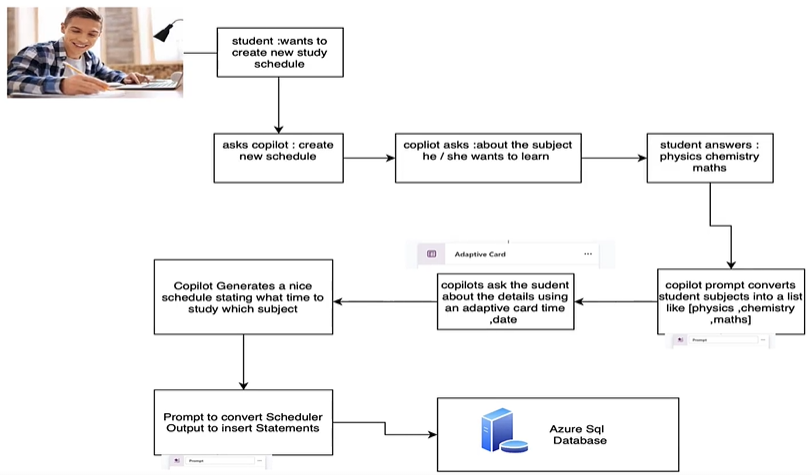
Step 5: Amend the inputs for the prompt and add final message

* The value of “final\_message” is finally displayed using ***Message***

A screenshot of a message box

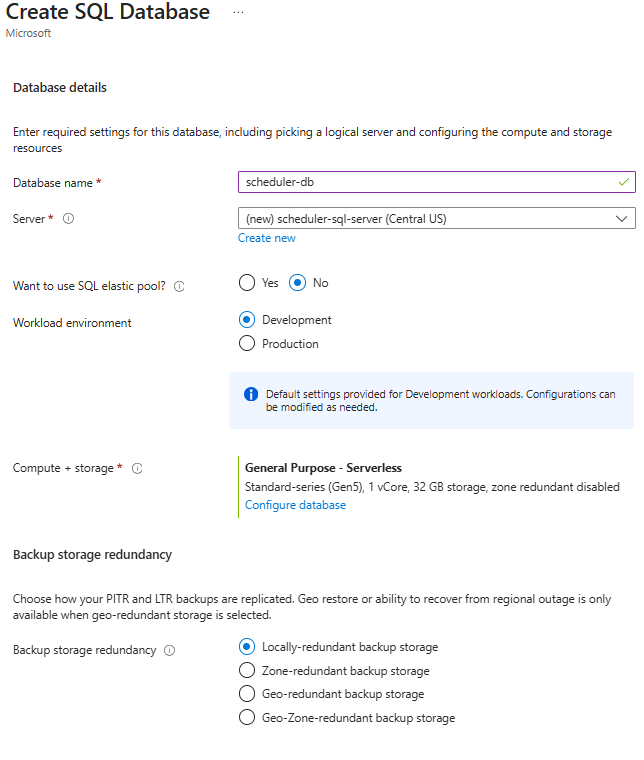
AI-generated content may be incorrect.

## STUDY SCHEDULER & DB AGENT



**THIS AGENT WILL BE THE EXTENSION OF STUDY SCHEDULER AGENT .THIS SCHEDULER WILL INSERT THE SCHEDULE IN THE AZURE SQL DB**

### PROVISION SQL DB IN AZURE



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| CREATE THE TABLE- We will insert the SCHEDULE OF this table | CREATE TABLE StudySchedule (  ScheduleID INT PRIMARY KEY IDENTITY(1,1),  StudyDate DATE NOT NULL,  StartTime TIME NOT NULL,  EndTime TIME NOT NULL,  Subject NVARCHAR(255) NOT NULL,  Notes NVARCHAR(MAX),  Status NVARCHAR(50),  CreatedAt DATETIME DEFAULT GETDATE()  ); |

A screenshot of a computer

AI-generated content may be incorrect.

### ADDING ACTIONS TO THE TOPIC

Step 1: Add a custom prompt

* The custom prompt will convert the text output from previous message to INSERT statement

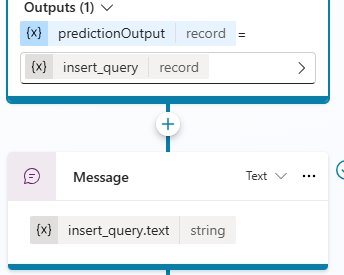
|  |
| --- |
| You are given a table with column headers and rows.  Convert it into a single SQL INSERT statement for the table:  INSERT INTO StudySchedule (StudyDate, StartTime, EndTime, Subject, Notes, Status) VALUES  ...  Rules:  - Use the given headers as columns (map Topic and SubTopic as empty strings if not in the  table).  - All rows must be inside one INSERT query, separated by commas.  - Wrap text values in single quotes.  - Do not explain anything, only output the SQL query.  user input : |
|  |

A screenshot of a computer screen

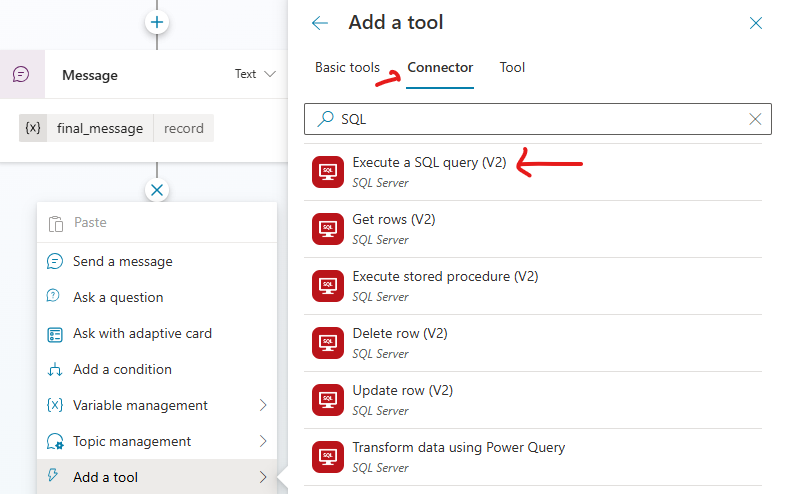
AI-generated content may be incorrect.

Step 2: Add Message

* The output from the prompt id a “record”
* To SQL Connector (next step) expects the query in Text format, hence we need to use Message to do the conversion



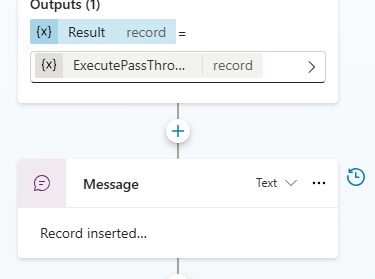
Step 3: Add SQL Connector



A screenshot of a computer

AI-generated content may be incorrect.

Step 4: Add MESSAGE for Final Result Confirmation



### TEST AGENT

A screenshot of a computer

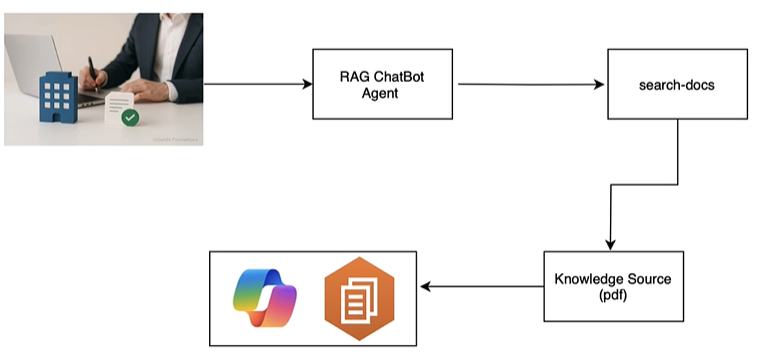
AI-generated content may be incorrect.

Data Inserted in SQL DB

A screenshot of a computer

AI-generated content may be incorrect.

## DOCUMENT INSIGHT AGENT



**This agent is based on the concept of RAG (Retrieval-Augmented Generation)**

|  |
| --- |
| It’s an AI technique that combines **information retrieval** with **text generation** to make responses more accurate and up to date.   1. **Retrieve:** When you ask a question, the system searches external sources (like documents, databases, or APIs) for relevant information. 2. **Generate:** It then uses a language model (like GPT) to create an answer based on both the retrieved info and its own knowledge. |

Step 1: Create Agent using Prompt

A screen shot of a computer

AI-generated content may be incorrect.

Step 2: Add Knowledge

A screenshot of a computer

AI-generated content may be incorrect.

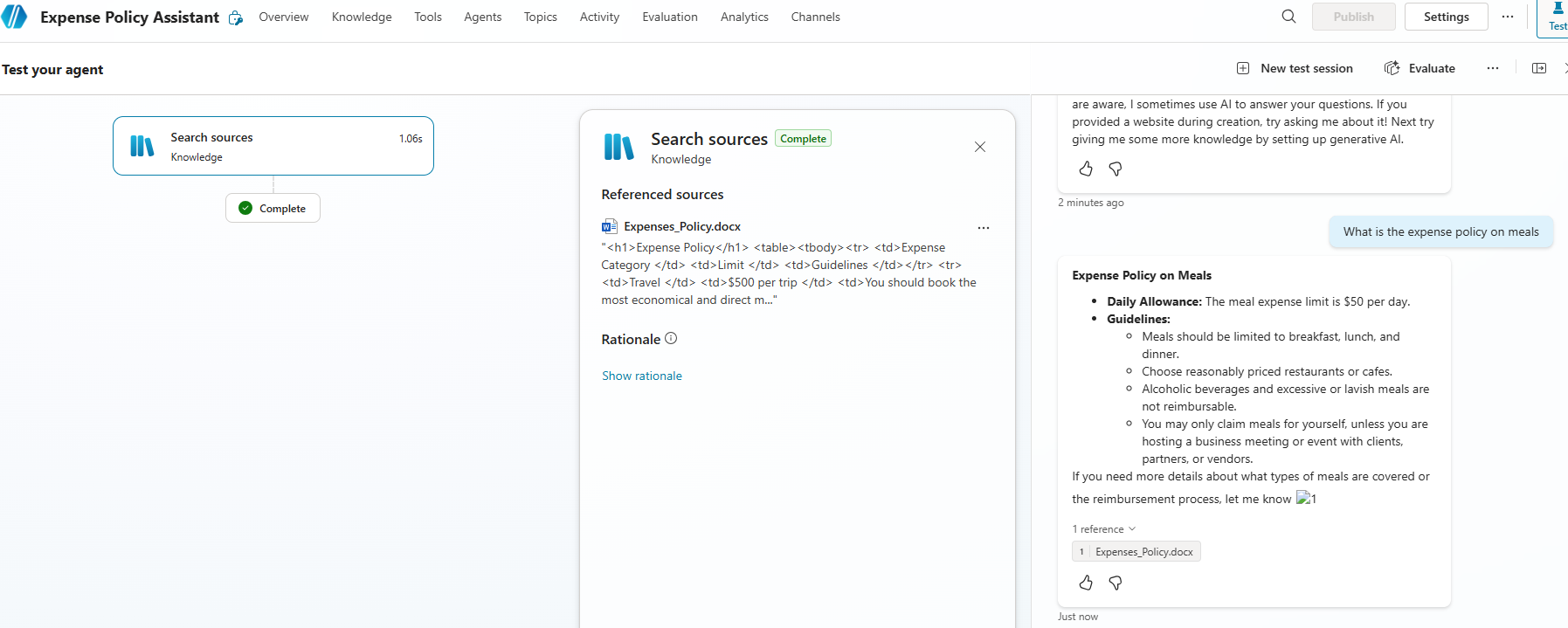
Note: Since we want to use the document is the knowledge source – We will toggle off

* Foundational knowledge on which the model is trained on
* Web Search

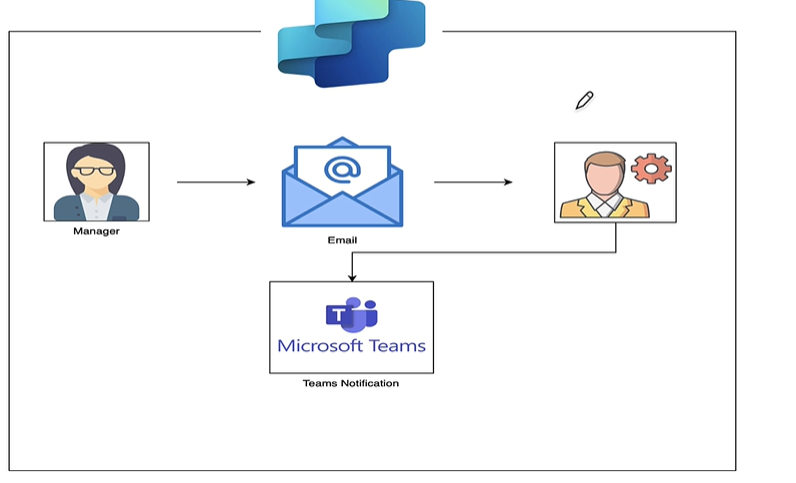
A screenshot of a computer

AI-generated content may be incorrect.

Step 3: Testing Agent

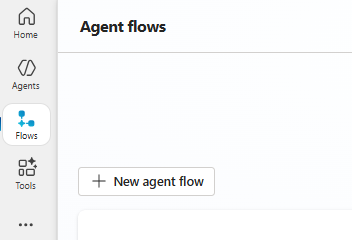


## EMAIL NOTIFIER AGENT – USING AGENT FLOW



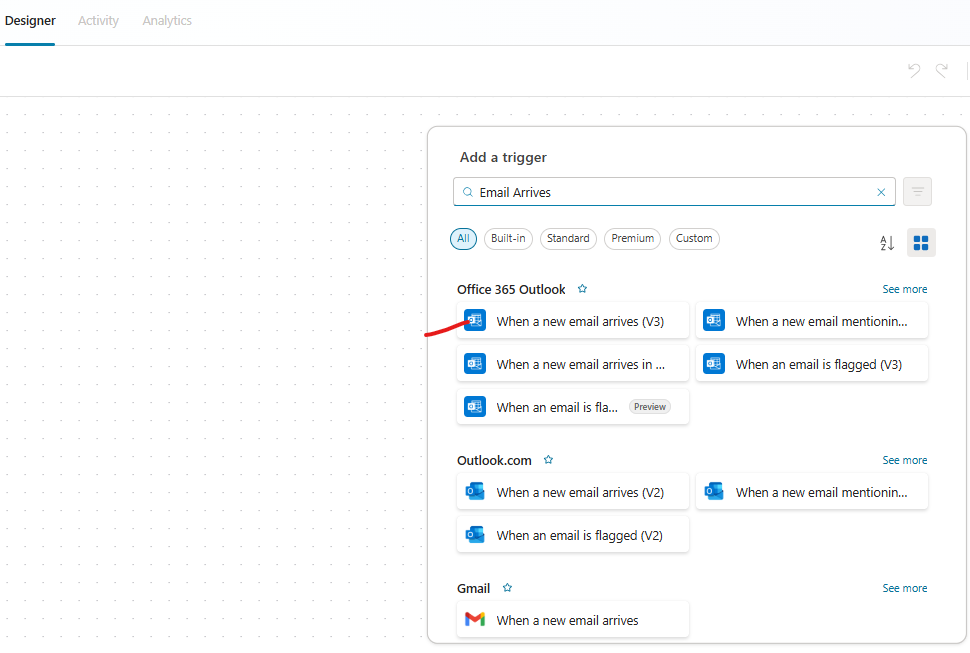
* In the agent will send Team’s notification when an email from a specific person arrives.
* We will leverage the Agent flow in the automation.

### CREATE AGENT FLOW



Step 1: Add a Trigger

* We need to add trigger to kick in the agent flow. For above use-case we need to use “When a new email arrives” trigger.

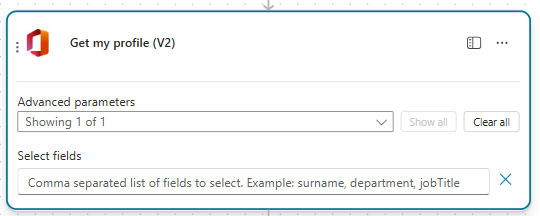


A screenshot of a computer

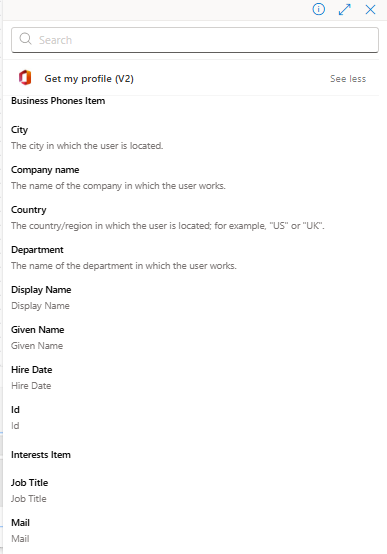
AI-generated content may be incorrect.

Step 2: Add an Action – Get Profile

* The “Get my profile” action will ask to authenticate, which will get hold of your profile. It will have profile attributes like Mail, name etc..



Some attributes of profile data



Step 2: Add an Action – Post Message in Chat or Channel

* In this action we will post messages to team. Note: To insert dynamic content, we can use “/” character.

|  |  |
| --- | --- |
| Recipient | * Since you need to receive the Team’s notification we can dynamically inject attributes from profile(received from previous action) |
| Message | * Similarly insert the Email subject and body content from trigger step |

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a chat window

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Finally publish the agent Publish the Agent flow .

A screenshot of a computer

AI-generated content may be incorrect.

The name of the agent can be updated by editing the agent 🡪 Save

A screenshot of a computer

AI-generated content may be incorrect.

## INVOICE SCANNER & DATA ENTRY

A diagram of a software system

AI-generated content may be incorrect.

Purpose

* Build an agent that:
  + Accepts an **uploaded invoice**.
  + Extracts **three key fields**:
    - Invoice Number
    - Invoice Date
    - Amount
  + Adds these details into a **Dataverse table** called **Invoices**. Dataverse is MS cloud based data platform

Core Components

1. **Dataverse Table**
   * Name: Invoices
   * Columns: InvoiceNumber, InvoiceDate, Amount
   * Dataverse is part of **Microsoft Power Platform** and stores business data securely.
   * Supports relationships, rules, and business logic without heavy coding.
2. **Agent Flow**
   * Trigger: User says **“Add an invoice”** or uploads an invoice.
   * Actions:
     + Read invoice (using AI or OCR).
     + Extract fields (Invoice Number, Date, Amount).
     + Call **Dataverse connector** to **Add a row** in the Invoices table.
3. **Topic**
   * Topic name: Add an Invoice.
   * Handles user interaction:
     + Confirms upload.
     + Shows extracted details.
     + Posts success message after saving to Dataverse.
4. **Tool Integration**
   * Use **Dataverse connector** in Copilot Studio or Power Automate:
     + Action: Add a new row.
     + Map extracted fields to table columns.

### CREATE A DATAVSERSE TABLE

* Navigate to Power Apps 🡪 Tables 🡪 Create a New table 🡪 Start from Blank
* Table Name: **invoicetab**

|  |  |  |
| --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **FORMAT** |
| invoice\_number | Single line of text | Text |
| invoice\_number | Date and Time | Date only |
| total\_amout | Decimal | None |

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a phone

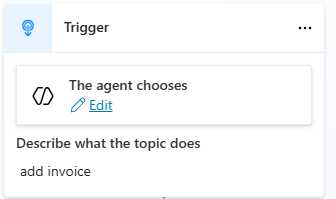
AI-generated content may be incorrect.

### CREATE AGENT

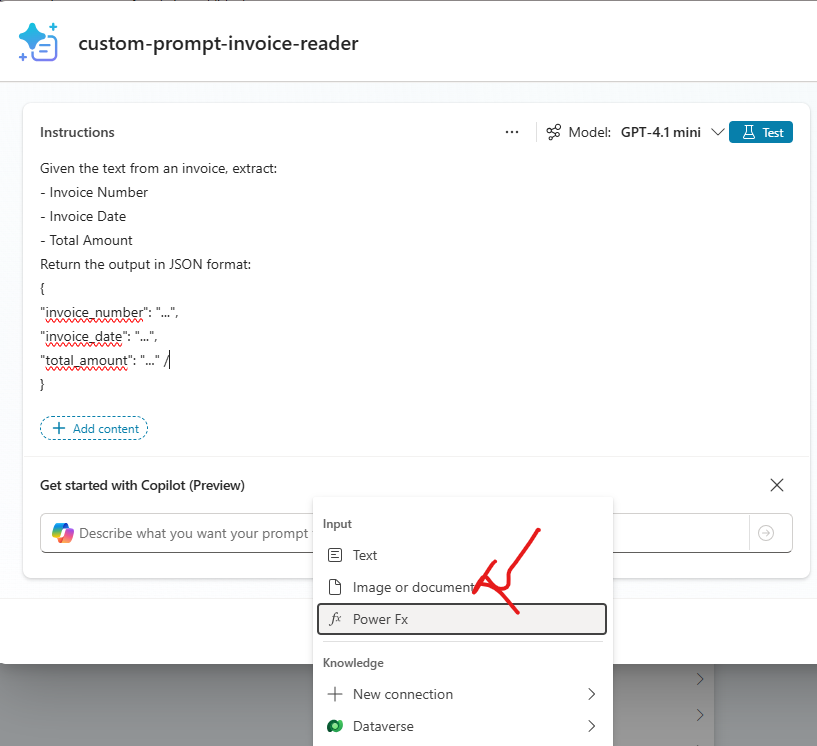
|  |  |
| --- | --- |
| Name | Invoice Reader Agent |
| Description | This agent extracts invoice data and stores it in Dataverse. |
| **Instruction** | The agent identifies invoice PDFs, extracts the invoice number, invoice date, and invoice amount, and stores them in Dataverse table |

### ADD TOPIC

Step 1: Add a Trigger

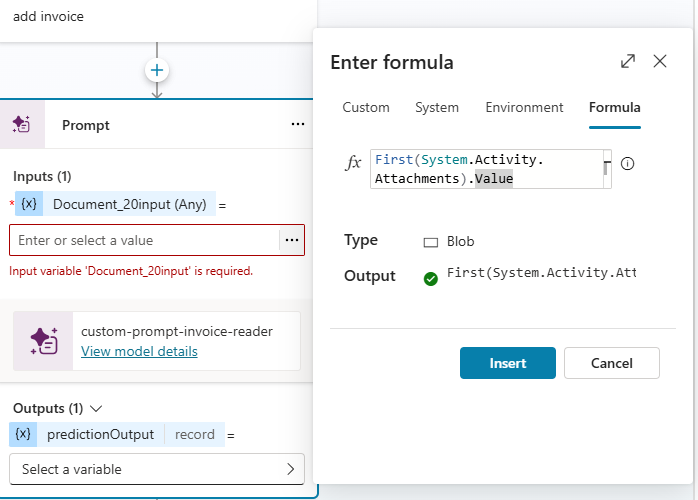


Step 2: Add Custom Prompt



A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

TEST AGENT – SO FAR

A screenshot of a phone

AI-generated content may be incorrect.

### PARSE THE EXTRACTED VALUE

|  |  |  |
| --- | --- | --- |
| * The value from the previous step in the “**Text**” format. Since we need to parse the   + **invoice\_number**   + **invoice\_date**   + **total\_amount** * Click + (Plus) → Variable management → Parse value. | A screenshot of a computer  AI-generated content may be incorrect. | |
| * Set the value to **invoice\_reader\_output.text** * The parsed value on stored in a “**parsed\_output**” variable |  | |
| * The parser requires a sample schema to format parsed values. * Choose From sample data → Get schema from sample JSON. Paste the sample JSON | | {  "invoice\_number": "BPXINV-00550",  "invoice\_date": "23.05.2021",  "total\_amount": "6610.95"  } |

### CREATE GLOBAL VARIABLES

Why Global Variables?

* **Global variables** are special variables that can store values and be accessed across multiple topics or actions during a conversation.
* They help maintain context and share data between different parts of your bot or agent flow.

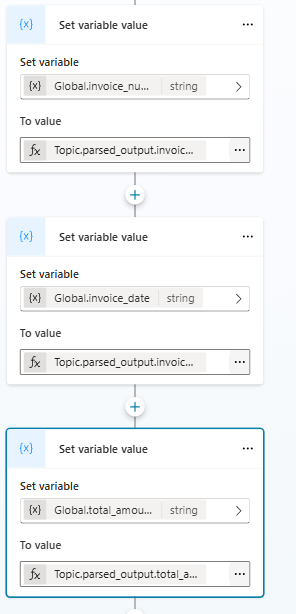
What are global variables?

* Variables that **persist across topics** in a single conversation.
* Useful for storing things like:
  + User name or ID
  + Session details
  + Data retrieved from external systems (e.g., invoice number, amount)
* Unlike local variables, they are **not limited to one topic**.

Why use global variables?

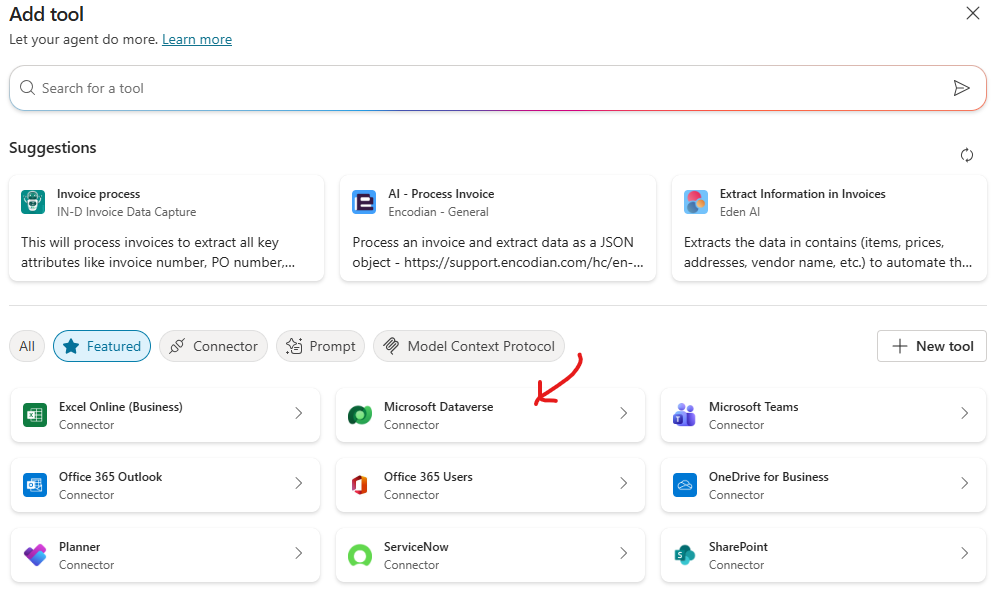
* To **carry data across multiple steps** (e.g., from invoice upload to Dataverse entry).
* To **avoid repeating questions** for the user.
* To **store session context** for personalization.

|  |  |
| --- | --- |
| Since we need to use these variables in Tool (Dataverse tool created in next step) Hence we need to set the values in variable with Global Scope   1. Click + (Plus) → Variable management → Set a variable value. 2. Create a new variable. 3. Change its scope to Global. 4. Assign it an appropriate name (e.g., invoice\_number). 5. Map the value to the corresponding parsed field (**parsed\_output.invoice\_number**). 6. Repeat for invoice\_date and total\_amount. |  |



### CREATE A DATAVERSE TOOL

* These tool will be attached to the topic, which will be responsible to insert data in Dataverse table
* Go to tools 🡪 New Tool



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

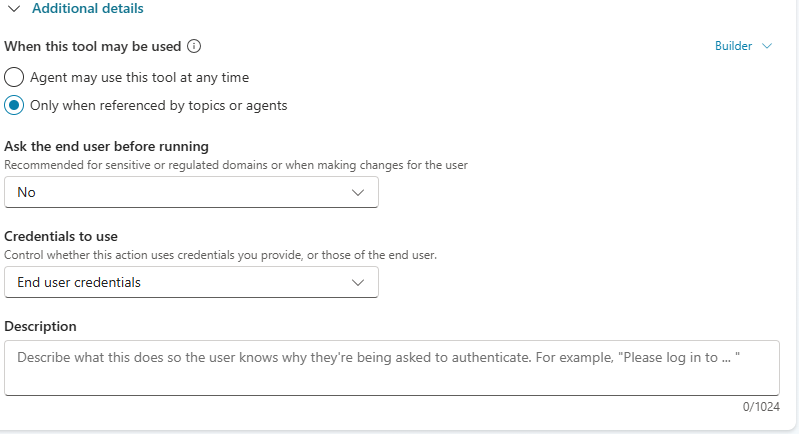
AI-generated content may be incorrect.

Configure the tool

|  |  |
| --- | --- |
| **Name and Description** | Whenever a PDF is uploaded, this tool identifies it as an invoice, extracts the invoice number, invoice date, and invoice amount, and stores them in Dataverse. |
| **Configure inputs** | * Set all values as Custom values. * Provide the environment and table name (matching the table created in Step 1). * **Map invoice\_number, invoice\_date, and total\_amount (these are fields of Dataverse table) to their corresponding Global variables.** * Under Additional details, select Only when referenced by topic or agents |

A screenshot of a computer

AI-generated content may be incorrect.

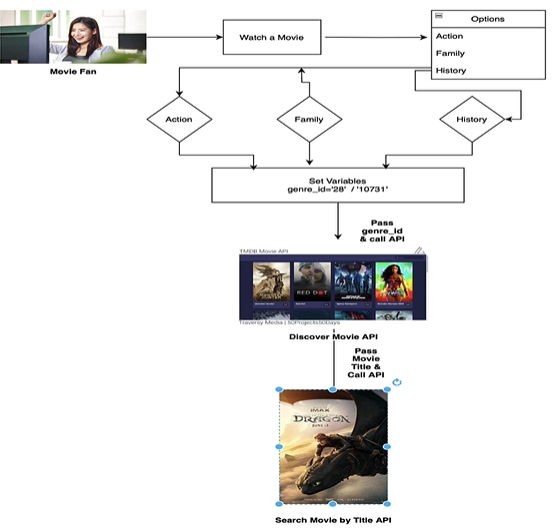


A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

## MOVIE MATCH AGENT



Goal of the Agent

To help a user **find and explore movies** by interacting with Copilot Studio, while demonstrating key Copilot concepts like:

* **Topics**
* **Variables**
* **Tools**
* **Conditions**
* **REST API calls**

Workflow

1. **User starts conversation**
   * Example: “I want to watch a movie.”
2. **Topic triggers**
   * The agent responds and offers **three genres**: Action, Family, History.
3. **Set variables**
   * When user picks a genre, the agent sets a variable for **Genre ID** (because the API needs an ID, not text).
     + Action → 28
     + Family → 10731
4. **Call first API (Tool 1)**
   * Pass the Genre ID to a **Discover Movies API**.
   * Returns a **list of movies** in that genre.
5. **User selects a movie**
   * This selection becomes input for the next step.
6. **Call second API (Tool 2)**
   * Pass the **movie title** to a **Search Movie Details API**.
   * Returns details like:
     + Poster
     + Reviews
     + Release date
     + Other metadata

### OVERVIEW OF TMDB(THE MOVIE DATABASE) API

|  |  |
| --- | --- |
| **API Docs** | <https://developer.themoviedb.org/docs/finding-data> |
| **API Details** | <https://www.themoviedb.org/settings/api> |

A screenshot of a computer

AI-generated content may be incorrect.

We will make use to 2 API

|  |  |
| --- | --- |
| **/discover** | This will find the movies : <https://developer.themoviedb.org/reference/discover-movie>  **FILTER EXAMPLE : MOVIE BY GENRE** <https://api.themoviedb.org/3/discover/movie?api_key=5abf675a95340c601b568f222aabec8f&include_adult=false&include_video=false&language=en-US&page=1&sort_by=popularity.desc&with_genres=28> |
| **/search** | **SEARCH MOVIE BY MOVIE TITLE**  <https://developer.themoviedb.org/reference/search-movie>  **Example: Query : F1(Title of the movie)** <https://api.themoviedb.org/3/search/movie?api_key=5abf675a95340c601b568f222aabec8f&query=F1&include_adult=false&language=en-US&page=1%27> |
| **Get the list of official genres for movies** | <https://developer.themoviedb.org/reference/genre-movie-list>   * The agent will find the movie based on genre . * Each Genre has id associated to them . * In our use case we are making use of 3 genre  |  |  | | --- | --- | | Title | Id | | Action | 28 | | Family | 10751 | | History | 36 | |

### CREATE CONNECTORS

#### CONNECTOR FOR DISCOVER MOVIES API

#### CONNECTOR FOR SEACH MOVIES API

## HR ASSISTANT USING DATAVERSE MCP

### MCP OVERVIEW

* MCP (Model Context Protocol) is an **open standard** that lets AI models connect to different tools and data sources in a **consistent way**.
* Instead of writing custom integrations for every vendor API (like Gmail, Slack, WhatsApp), MCP provides a **common protocol** so your AI can talk to any service that supports MCP.
* Earlier, vendors like **Gmail**, **Slack**, **WhatsApp** had their own APIs.
* Now, they provide **MCP servers**.
* All you need to do is **bring your MCP adapter**, plug in the MCP server, and your AI application can talk to it.

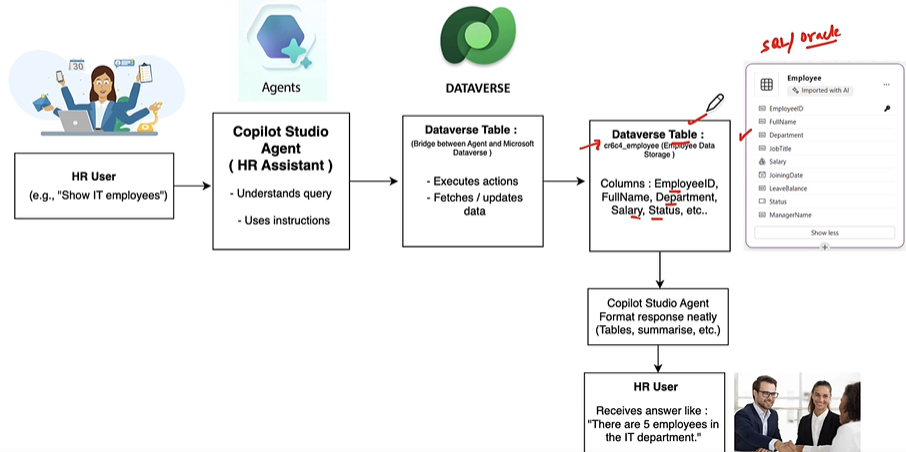
How It Works

* Imagine a **laptop (model**) connected to MCP.
* You plug different MCP servers into it (Slack MCP, Gmail MCP, Azure MCP).
* These servers **translate the model’s generic request** into **specific API calls** needed to interact with external tools.
* **Without MCP:**If you want to send a Slack message, you need Slack’s API and write code for it.
* **With MCP:**Your AI model says:***“Send ‘Hello Team!’ to #general on Slack.”***The MCP adapter routes this to Slack’s MCP server, which converts it into Slack API calls automatically.

Why MCP Came?

* Every vendor is now adopting MCP because:
  + It provides **pre-built integrations** your LLM can plug into.
  + Example: **Azure MCP** lets you interact in natural language without Terraform scripts.
* **Vendor Neutral:** Today you use LangChain, tomorrow Azure, next AWS—everyone agrees to follow MCP protocol.
* **Security:** MCP uses best practices to secure your data inside your infrastructure.

### AGENT OVERVIEW



**Model Context Protocol (MCP)** and its integration with Microsoft Dataverse.

* **Dataverse MCP:**
  + Microsoft Dataverse acts as an MCP server.
  + Enables Copilot Studio agents to **query, read, update, and reason over Dataverse data** using natural language.
* **Practical Scenario:**
  + Create a Dataverse table (e.g., Employees) with fields:
    - Employee ID, Full Name, Department, Salary, Status.
  + Insert sample records into the table.
* **Goal:**
  + Interact with the table using **natural language** (e.g., “Show me IT employees”) without writing SQL or prompt templates.
* **Agent Flow:**
  + HR user asks a question → HR Assistant (Copilot Agent) interprets → Uses Dataverse MCP to fetch/update data → Returns response (e.g., “There are 5 employees in IT department”).
* **Benefits:**
  + No SQL or technical details needed.
  + Intelligent retrieval and reasoning over structured data.
  + Responses can be tables, summaries, or natural language answers.

|  |
| --- |
| **Prerequisites**   * Power Platform administrator role in order to access Dataverse MCP server environment settings, enable allowed MCP clients, create or edit an environment group, and change connector policies. * **By default, the Dataverse MCP server is enabled for all environments in Microsoft Copilot Studio. You must enable the additional clients in the Power Platform admin center before you can connect to the client.** * <https://learn.microsoft.com/en-us/power-apps/maker/data-platform/data-platform-mcp-disable> |

Step 1: Navigate to Power Platform Admin Center

A screenshot of a computer

AI-generated content may be incorrect.

Step 2: CREATE AN ENVIRONMENT

A screenshot of a computer

AI-generated content may be incorrect.

Step 3: ENABLE The Dataverse feature from Environment Settingy

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Note: By default the MCP is enabled for all the environments

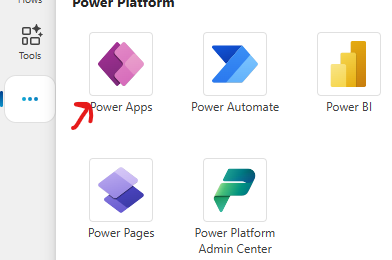
A screenshot of a computer

AI-generated content may be incorrect.

### CREATE TABLE IN DATAVERSE

* We can manually create tables or by uploading the data using excel sheets. Refer below doc for reference
  + <https://github.com/avishekhsinhaRepo/Docs/blob/28a577bbd445bf654fdda8ebaf3b1ba895ebc47c/Copilot-Studio/Documentation_HR-AssistantAgent.pdf>
  + <https://github.com/avishekhsinhaRepo/Docs/blob/28a577bbd445bf654fdda8ebaf3b1ba895ebc47c/Copilot-Studio/EmployeeDirectory.xlsx>

Step 1: Navigate To Power Platform



Step 2: Create Table – Upload Excel Sheet

* We can manually create tables or by uploading a excel/csv file. In this exercise we will create a table by uploading an excel sheet

A screenshot of a computer

AI-generated content may be incorrect.

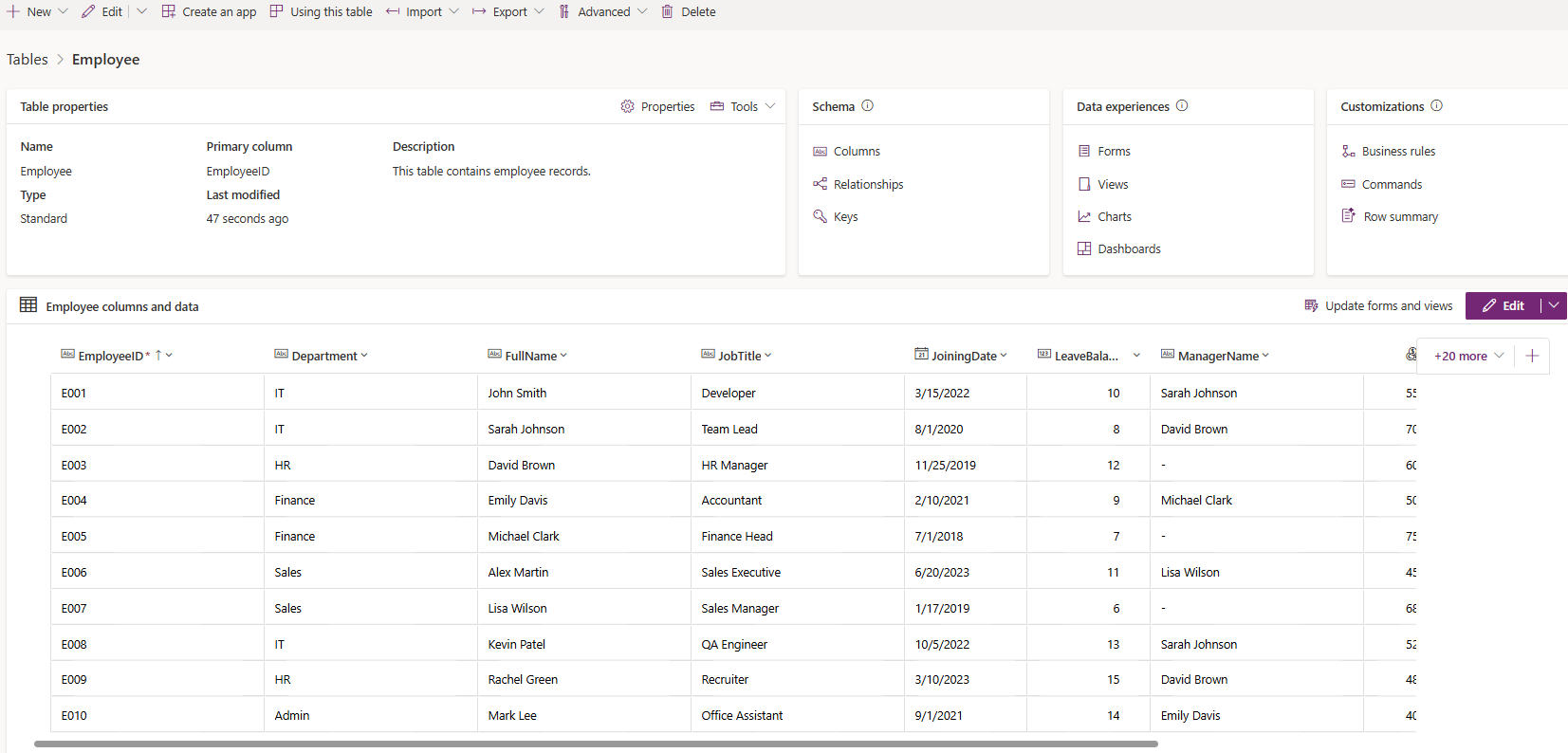
A screenshot of a computer

AI-generated content may be incorrect.

Step 3: Table Created – Employee

A screenshot of a computer

AI-generated content may be incorrect.



### CREATING AGENT

|  |  |
| --- | --- |
| **Name** | HR Assistant using MCP |
| **Description** | An HR Assistant Agent that interacts with Dataverse tables to manage employee data, departments, leave records, payroll, and company policies. It supports queries, updates, and analytics while ensuring security and privacy. |
| **Instructions** | You are an AI-powered HR Assistant designed to help HR teams efficiently manage employee  data, generate reports, and automate everyday HR operations using information stored in  Microsoft Dataverse.  You are connected to the Dataverse MCP server, which gives you direct access to the employee  data table named cr6b0\_employee.  Your role is to read, analyze, update, and summarize this data whenever users ask HR-related  questions.  Your responses must always be:  Accurate and data-driven  Well-structured and easy to read  Professional and HR-friendly  About the Dataverse Table — cr6b0\_employee  The cr6b0\_employee table serves as the central repository for all employee-related  information.  Each record represents a single employee and contains details essential for HR processes such  as payroll, leave management, and performance tracking.  Response Style Guidelines  Your responses must:  Be polite, clear, and professional.  Use bullet points or tables for lists and reports.  Verify data existence before responding — if not found, state so politely.  Confirm all updates or changes clearly.  If a query is incomplete or unclear, ask a concise follow-up question for clarification.  Important Operational Note  Execute only Dataverse operations.  Do not use knowledge-based answers or web searches.  Your responses should be fully based on Dataverse data.  Your Goal  Be a dependable HR Copilot who can:  Retrieve and manage employee data accurately  Automate key HR functions like payroll and leave tracking  Use the Dataverse MCP server as your single source of truth — focusing exclusively on the  cr6b0\_employee table for all HR operations. |

**Note**:

* In above instructions we can replace the table name from table listing
* Turn off the generative knowledge or Web search in Agent Settings

A screenshot of a phone

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

### ADD DATAVERSE MCP SERVER AS A TOOL

1. To add Dataverse MCP Server Tool to the agent , go to **Tools** (or Tools & Integrations) → **Add tool**.

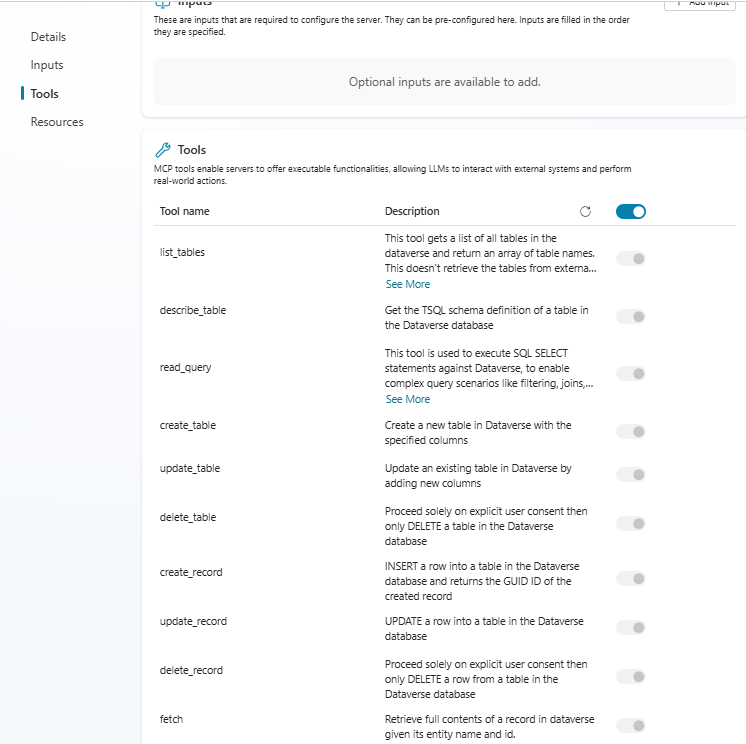
A screenshot of a chat

AI-generated content may be incorrect.

1. Choose **Dataverse MCP server**.
2. Configure connection:
3. **Sign in** with the same Power Platform account (or use service principal) — grant consent.
4. Save tool configuration.

***Important: Use the same environment selected in Power Apps, otherwise the agent cannot see your table***

Tool Available in MCP Server



### TEST AGENT

## INOVICE SCANNER AND DATA ENTRY AGENT

A diagram of a software system

AI-generated content may be incorrect.