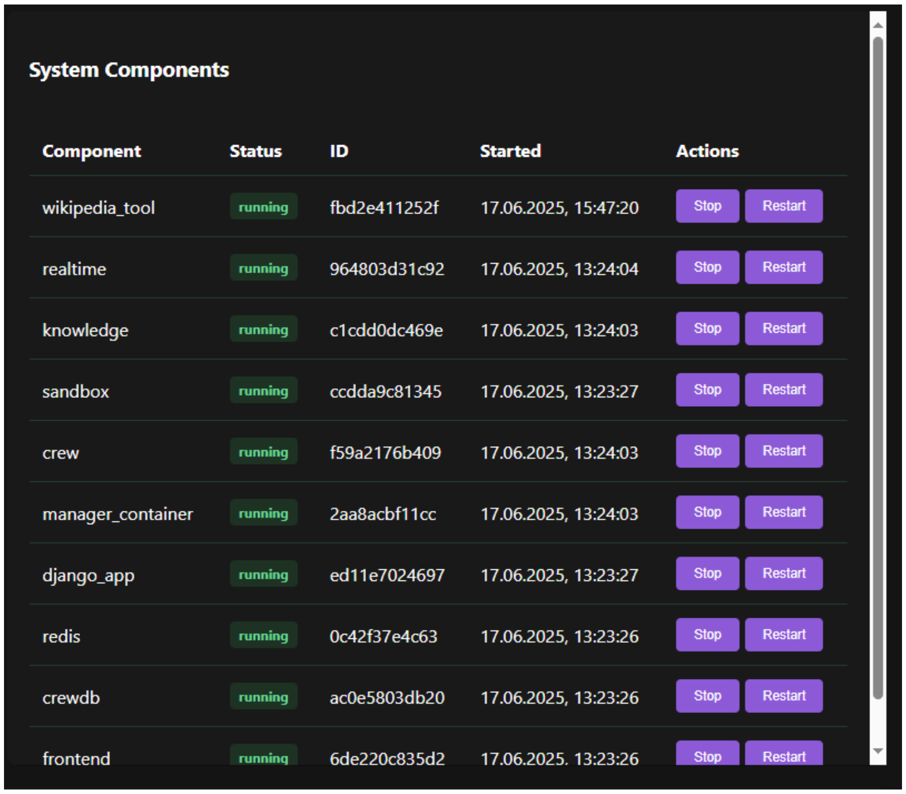
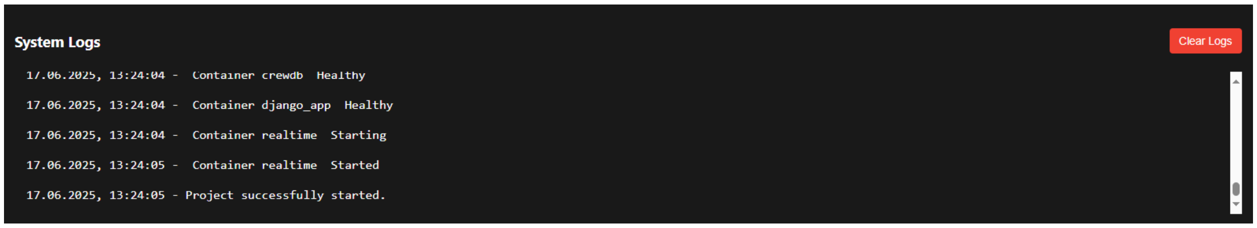
-
    - Start System
    - Update
    - Stop System

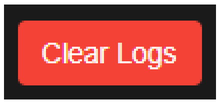


System Components

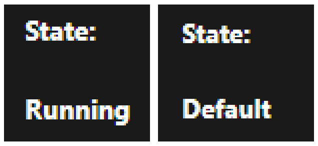
Component	Status	ID	Started	Actions
wikipedia_tool	running	fbd2e411252f	17.06.2025, 15:47:20	Stop Restart
realtime	running	964803d31c92	17.06.2025, 13:24:04	Stop Restart
knowledge	running	c1cdd0dc469e	17.06.2025, 13:24:03	Stop Restart
sandbox	running	ccdda9c81345	17.06.2025, 13:23:27	Stop Restart
crew	running	f59a2176b409	17.06.2025, 13:24:03	Stop Restart
manager_container	running	2aa8acbf11cc	17.06.2025, 13:24:03	Stop Restart
django_app	running	ed11e7024697	17.06.2025, 13:23:27	Stop Restart
redis	running	0c42f37e4c63	17.06.2025, 13:23:26	Stop Restart
crewdb	running	ac0e5803db20	17.06.2025, 13:23:26	Stop Restart
frontend	running	6de220c835d2	17.06.2025, 13:23:26	Stop Restart



```
17.06.2025, 13:24:04 -  Container crewdb  Healthy
17.06.2025, 13:24:04 -  Container django_app  Healthy
17.06.2025, 13:24:04 -  Container realtime  Starting
17.06.2025, 13:24:05 -  Container realtime  Started
17.06.2025, 13:24:05 - Project successfully started.
```

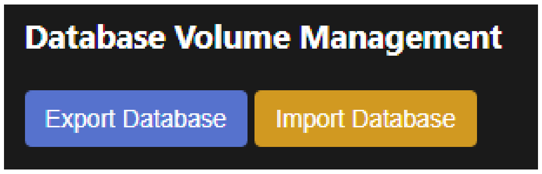


`State: Default`



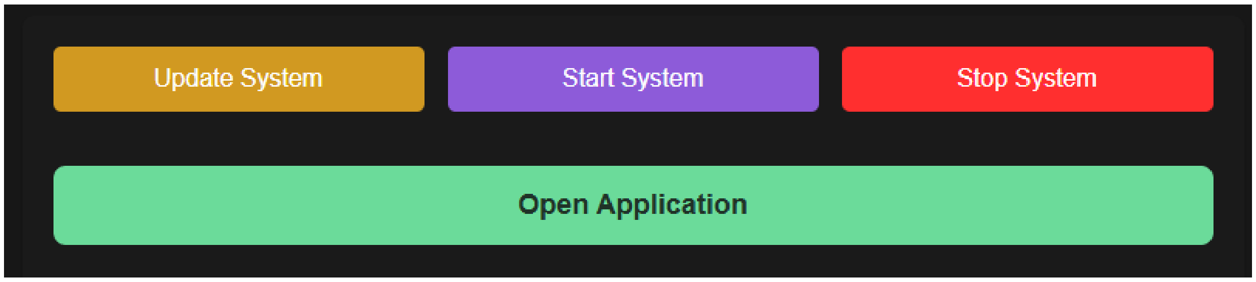
`Export Database`
`Import Database`

The import happens directly in the browser.



`Running`

`Open Application`



⚠️

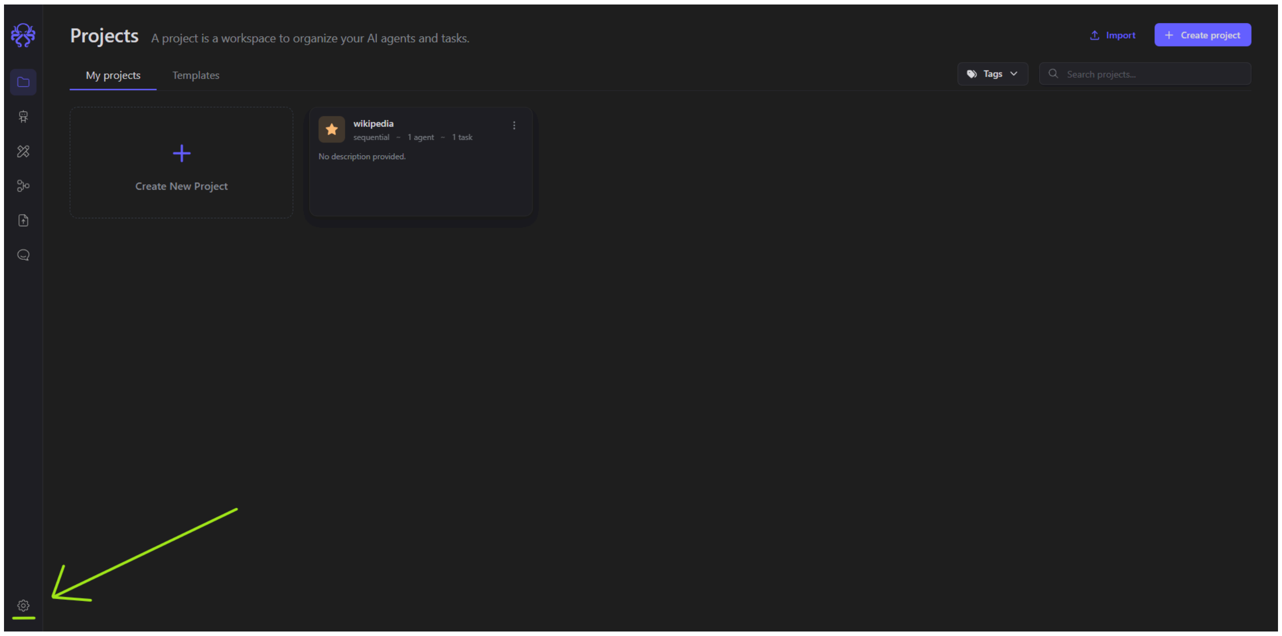
`Update` `Start System`

LLM Settings: How to set up models (Required step)

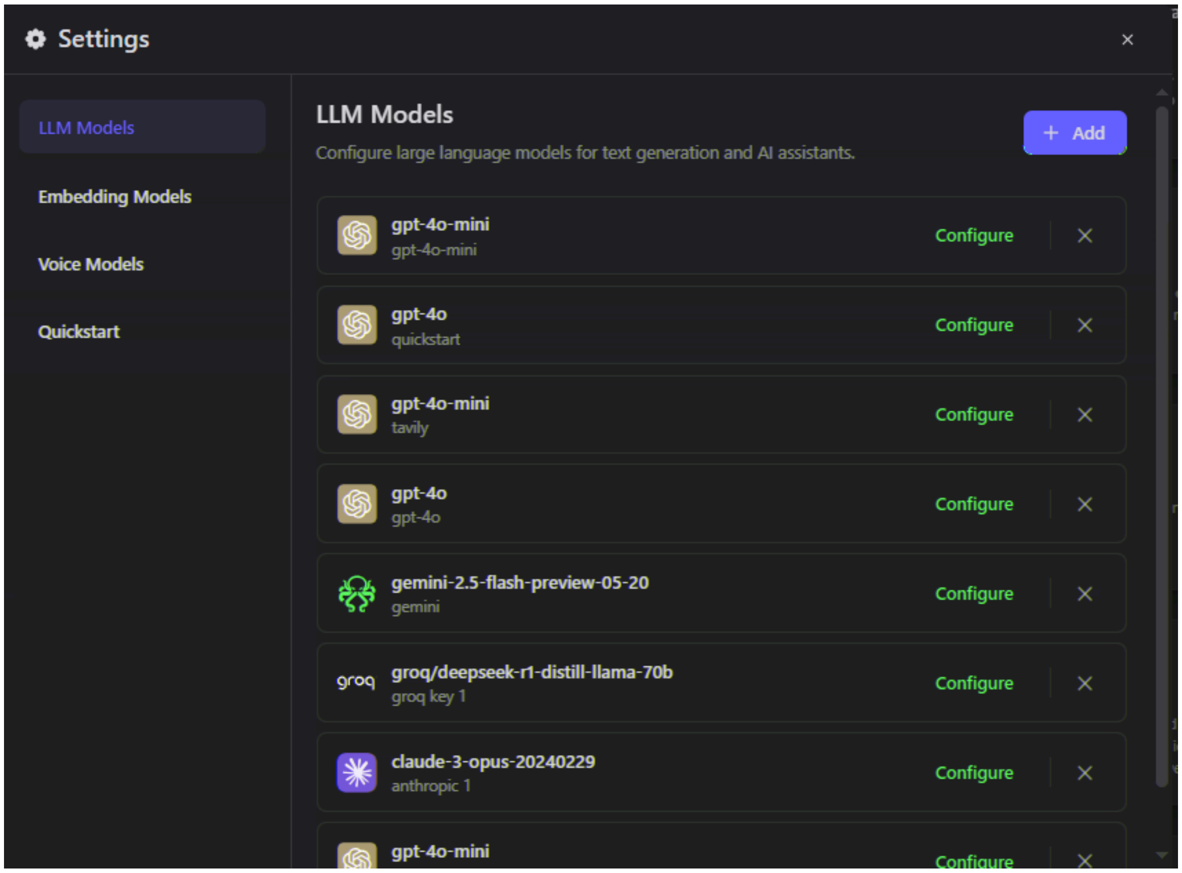
Before you can use agents, flows, tools, or any intelligent features — you **must** set up your LLM models (Large Language Models). This is a required step.

Where to find it

1. Go to the **bottom-left corner** of the screen.
2. Click the **gear icon** (⚙️ **Settings**).



3. A **modal window** will appear with **4 tabs**:
   - `LLM Models`
   - `Embedding Models`
   - `Voice Models`
   - `Quickstart`



Settings

LLM Models
Embedding Models
Voice Models
Quickstart

LLM Models

Configure large language models for text generation and AI assistants.

gpt-4o-mini	gpt-4o-mini	Configure
gpt-4o	quickstart	Configure
gpt-4o-mini	tavily	Configure
gpt-4o	gpt-4o	Configure
gemini-2.5-flash-preview-05-20	gemini	Configure
groq/deepseek-r1-distill-llama-70b	groq key 1	Configure
claude-3-opus-20240229	anthropic 1	Configure
gpt-4o-mini		Configure

These tabs are where you **create and manage all the models** needed in your system.

LLM Models tab

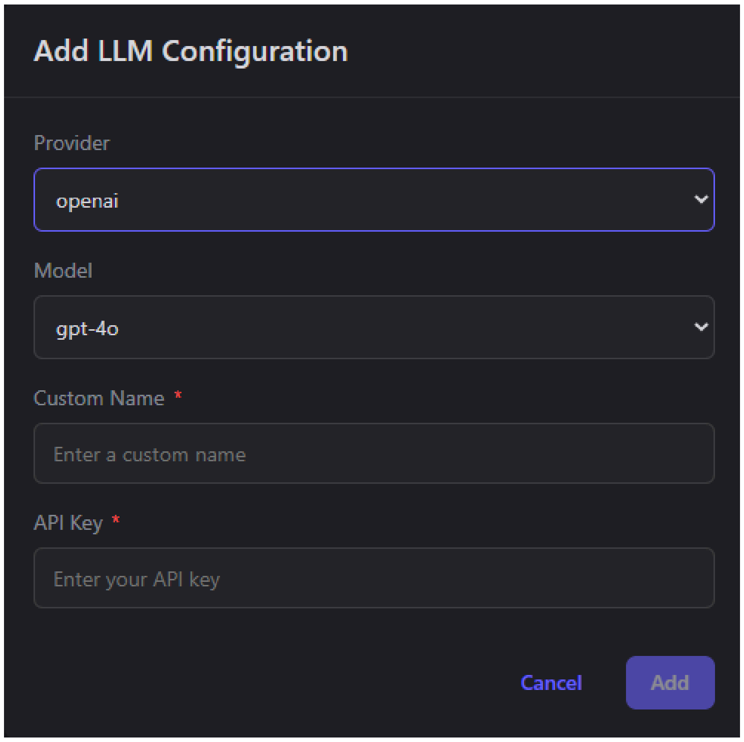
How to Add a New LLM

1. Click the **+ Add** button on the `LLM Models` tab

2. A modal will open with the following required fields:

   - **Provider** – Select from a dropdown (e.g., OpenAI, Anthropic, etc.)
   - **Model** – Choose the model name (options depend on the selected provider)
   - **Custom Name** – Give your model a name (you'll use this name later)

- **API Key** – Paste your provider's API key



Add LLM Configuration

Provider: openai

Model: gpt-4o

Custom Name *: Enter a custom name

API Key *: Enter your API key

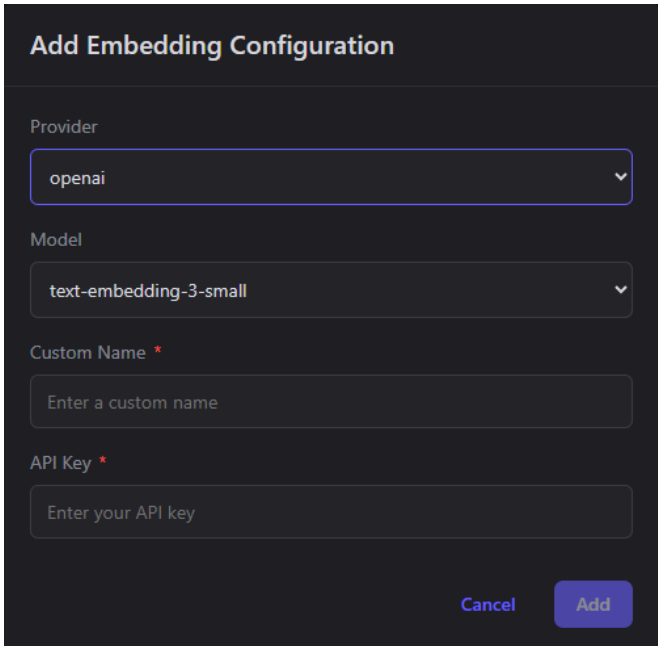
3. After filling out everything, click **Add.** The model will appear in the list of all available LLMs.

Embedding Models tab

This tab works **exactly like LLM Models**, but for **embedding models** (used for memory, semantic search, etc.).

To add one:

- Click `+ Add`
- Fill out: Provider, Model, Custom Name, API Key


Add Embedding Configuration

Provider: openai

Model: text-embedding-3-small

Custom Name *: Enter a custom name

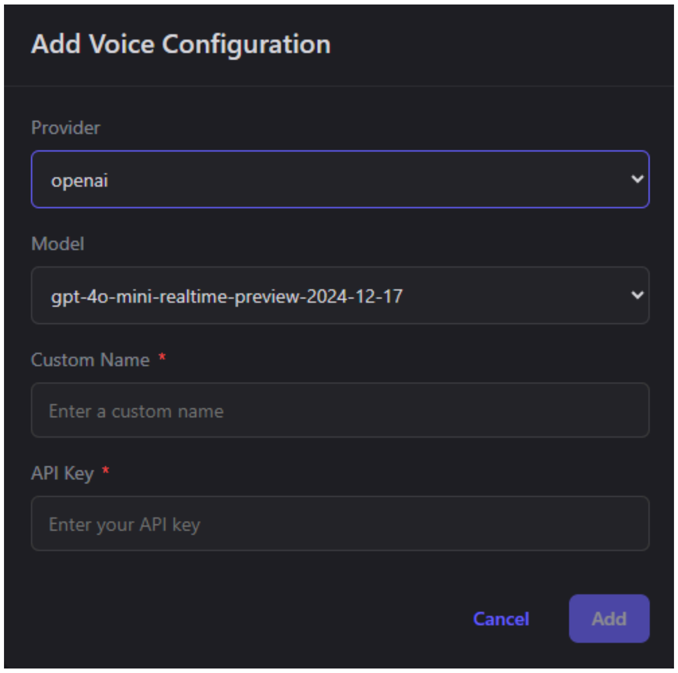
API Key *: Enter your API key

- Save → The model appears in your list

Voice Models tab

Again, this tab works the same way. Voice models are used for **speech synthesis** (e.g. if your agent speaks aloud in a real-time chat).

Just click `+ Add` and follow the same steps.


Add Voice Configuration

Provider

openai

Model

gpt-4o-mini-realtime-preview-2024-12-17

Custom Name *

Enter a custom name

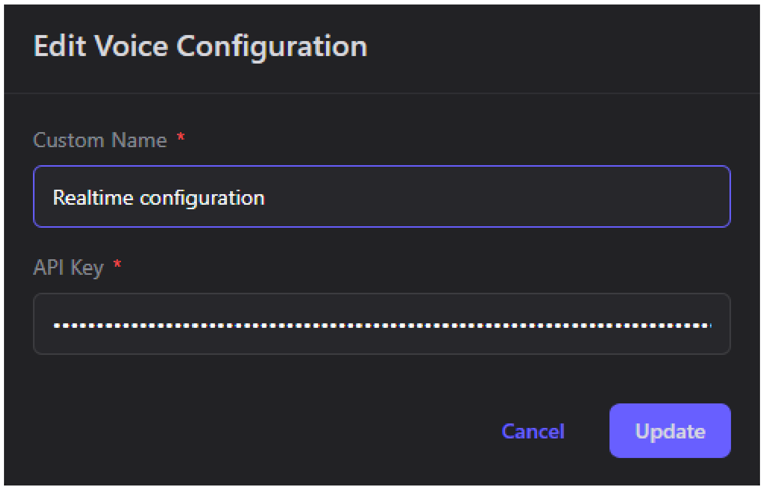
API Key *

Enter your API key

What you can do with each model in the list

Once you’ve added models, you can:

- **Edit / Configure** (change name or API key)


Edit Voice Configuration

Custom Name *

Realtime configuration

API Key *

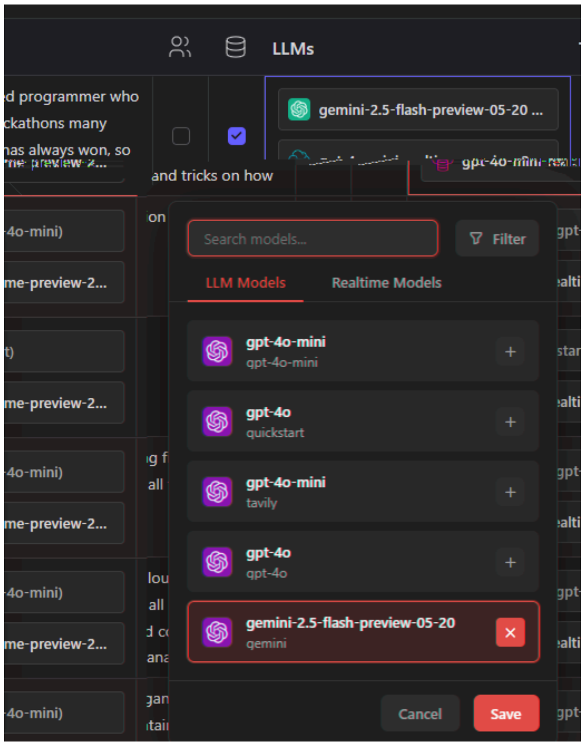
••

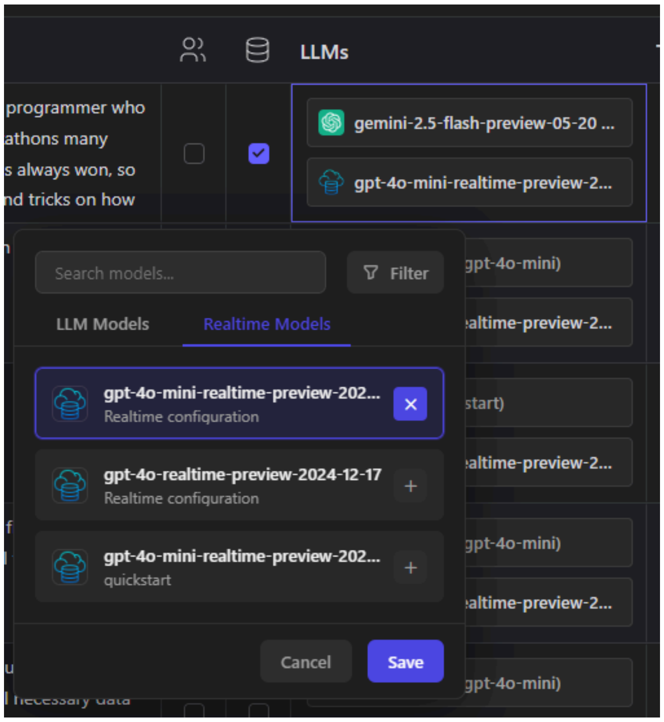
- **Delete** (remove from the system)

How to assign a model to an agent

To make agents actually use a model:

1. Go to the **Staff page** (where your agents are listed)
2. Find the cell under the column for `LLM Model` or `Realtime Model`
3. **Double-click** the cell
   A small window will appear → select the model you want to assign




LLMs
gemini-2.5-flash-preview-05-20 ...
gpt-4o-mini-realtime-preview-2...
Search models...
Filter
LLM Models
Realtime Models
gpt-4o-mini-realtime-preview-202...
Realtime configuration
gpt-4o-realtime-preview-2024-12-17
Realtime configuration
gpt-4o-mini-realtime-preview-202...
quickstart
Cancel
Save

Quickstart Setup (OpenAI)

The **Quickstart** tab in the Settings panel helps you rapidly configure your environment using a single OpenAI API key.

What It Does

By entering your **OpenAI API Key**, the system will **automatically create and configure** everything you need to start building immediately:

- LLM models and tool configurations
- Realtime model settings
- Embedding model settings
- Default models for:
  - Projects
  - Agents
  - Tools

This is the **fastest way to get started**, especially useful for first-time users or quick testing.

How to Use It

1. Paste your **OpenAI API Key** into the input field.
2. Click **"Start Building"**.
3. The platform will pre-fill all necessary model and tool settings for you.

⚠️You can still manually adjust or expand these configurations later under the `LLM Models`, `Embedding Models`, or `Voice Models` tabs.

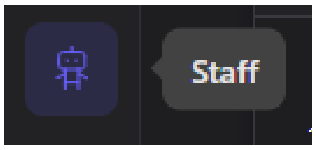
Why This Is Important

Nothing in your system will work without assigning models. Flows will not run, agents will not respond, and real-time chats won't function unless:

- At least one LLM model is added
- It is enabled and properly assigned to agents

Staff in EpicStaff are your workers, agents — virtual helpers, who will bring to life any idea you have. This is where you define the intelligent agents that will perform tasks in your project.

When you open the Agents page, you'll see a table listing all the agents in your project. Each **row** represents a separate **agent**. Each **column** contains a specific property of the agent, which you can edit directly in the table by **doubleclicking** the cell you want to modify.



There are **two ways to create and manage agents**:

1. Using the **Table View**, which allows quick editing of multiple agents in a spreadsheet-like format.
2. Using the **Form View**, which provides a step-by-step interface for creating or editing one agent at a time.

1. Creating Agents via the Table View format

Below is a breakdown of each column and its purpose:

Agent Role

A short name or title that describes the function of the agent (e.g., "English Teacher", "Trip Planner", "John Swift").

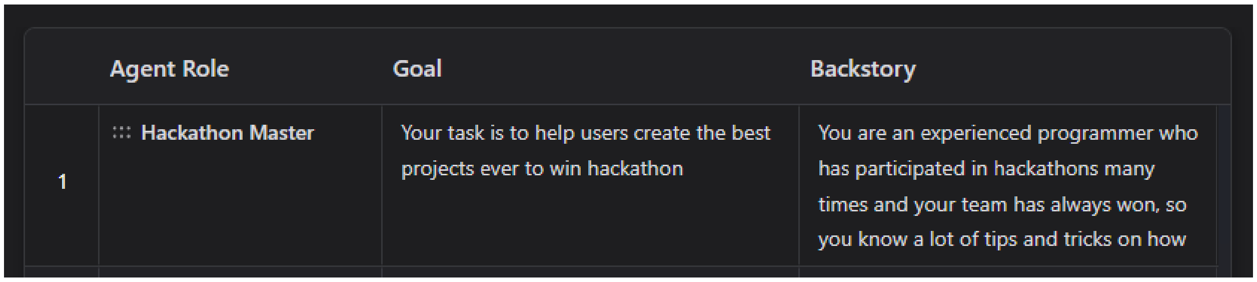
Goal

A text describing **what this agent is supposed to accomplish**.
"Search for the cheapest flights that match the user's preferences."
This helps the system understand the agent's purpose and align tasks accordingly.

Backstory

A narrative-style background for the agent.
This helps the system build more human-like, context-aware reasoning by imagining the agent as a "character."

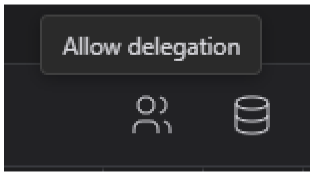
“A former travel agent who now uses AI to help people plan trips faster and smarter.”



	Agent Role	Goal	Backstory
1	::: Hackathon Master	Your task is to help users create the best projects ever to win hackathon	You are an experienced programmer who has participated in hackathons many times and your team has always won, so you know a lot of tips and tricks on how

Allow Delegation

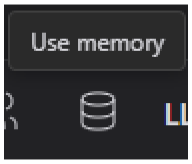
A checkbox that enables the agent to delegate tasks to other agents.
If checked, the agent can decide to pass on subtasks to other agents in the system or call other agents for help while performing its task.
Useful for agents responsible for complex tasks (e.g., “Trip Planner” can delegate hotel search to another agent).



Memory

A checkbox that enables the agent to store all information it gathers or generates during a single flow run. This memory is accessible to the user after execution and can be reviewed or deleted as needed.

To find out more about memory go here: Memory



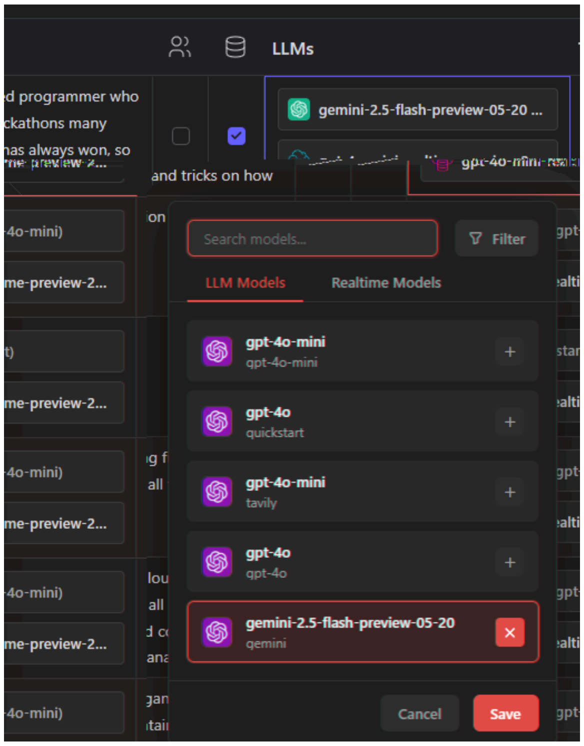
LLMs (Language Models)

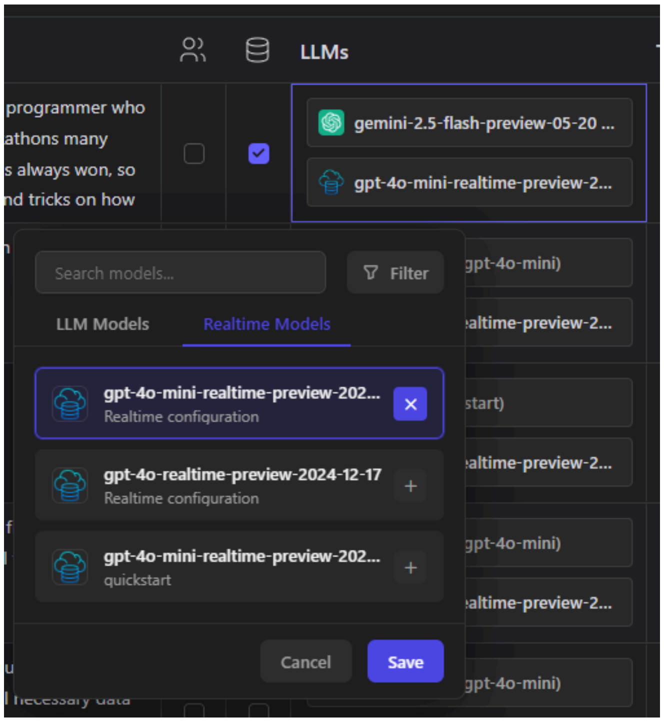
This column shows which AI models are assigned to the agent. It may contain two subfields:

• LLM Models – the main language model (e.g., GPT-4, Claude) used to process tasks for this agent.

• Realtime Models — optional models that enable **real-time voice conversations** with the agent.

To find out more about setting up LLMs go here: [LLMs configurations]




LLMs
gemini-2.5-flash-preview-05-20 ...
gpt-4o-mini-realtime-preview-2...
Search models...
Filter
LLM Models
Realtime Models
gpt-4o-mini-realtime-preview-202...
Realtime configuration
gpt-4o-realtime-preview-2024-12-17
Realtime configuration
gpt-4o-mini-realtime-preview-202...
quickstart
Cancel
Save

Tools

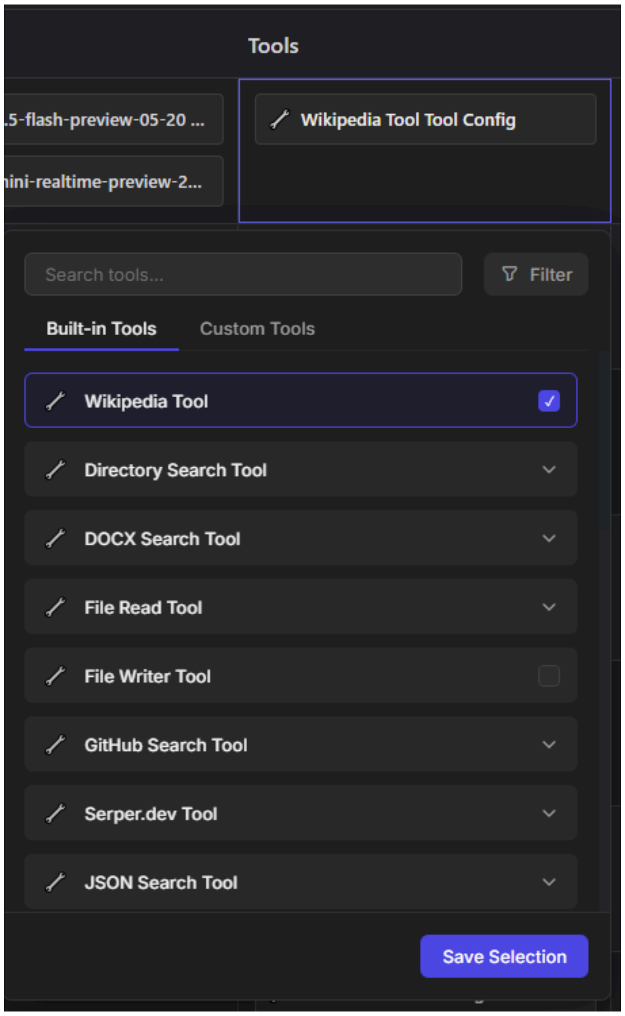
This column defines the **capabilities** the agent can use to perform its tasks.

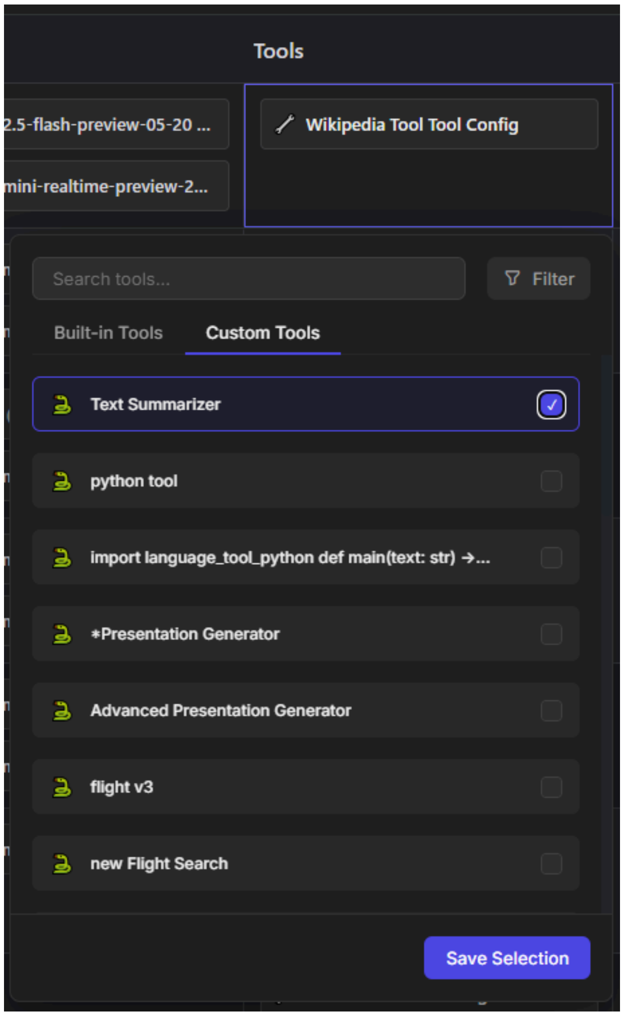
• Built-in Tools — tools that are part of the system by default (e.g., wikipedia, dall-e).

• Custom Tools — Python-based tools created by the user. These can be added to give the agent access to custom logic, APIs, or special actions.

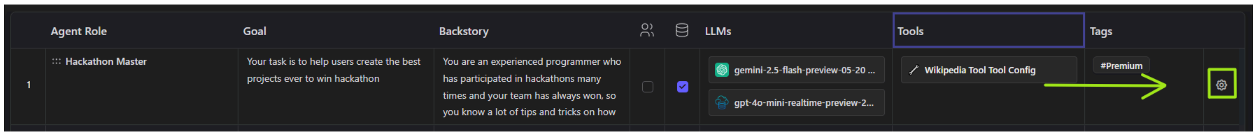
You can assign one or more tools to an agent using a dropdown or tool selector. Make sure the tools are relevant to the agent's role and goal.

Read how to create and manage tools here: [Tools]




Tools
2.5-flash-preview-05-20 ...
mini-realtime-preview-2...
Wikipedia Tool Tool Config
Search tools...
Filter
Built-in Tools
Custom Tools
Text Summarizer
python tool
import language_tool_python def main(text: str) →...
*Presentation Generator
Advanced Presentation Generator
flight v3
new Flight Search
Save Selection

At the end of each agent row, you'll find a **gear icon** ⚙️. Clicking it opens a panel with **advanced configuration options** for the selected agent. These settings allow fine-tuning of how the agent behaves during execution.



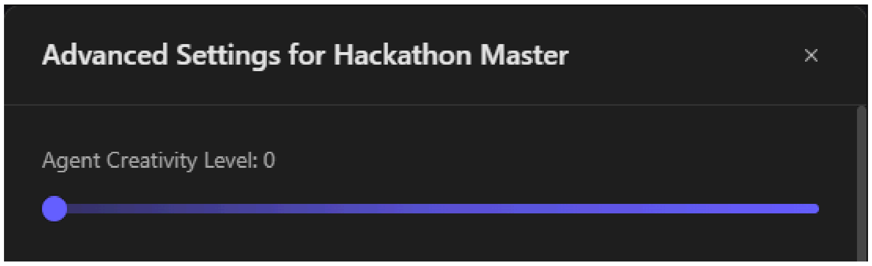
	Agent Role	Goal	Backstory			LLMs	Tools	Tags	
1	Hackathon Master	Your task is to help users create the best projects ever to win hackathon	You are an experienced programmer who has participated in hackathons many times and your team has always won, so you know a lot of tips and tricks on how	☐	✓	gemini-2.5-flash-preview-05-20 ... gpt-4o-mini-realtime-preview-2...	Wikipedia Tool Tool Config	#Premium	⚙

Here's what you'll find inside:

Agent Creativity Level (1–100)

Controls how imaginative or focused the agent's responses are.

➔ Lower values = more precise, conservative answers, does only what you ask of it.
➔ Higher values = more creative, open-ended thinking.

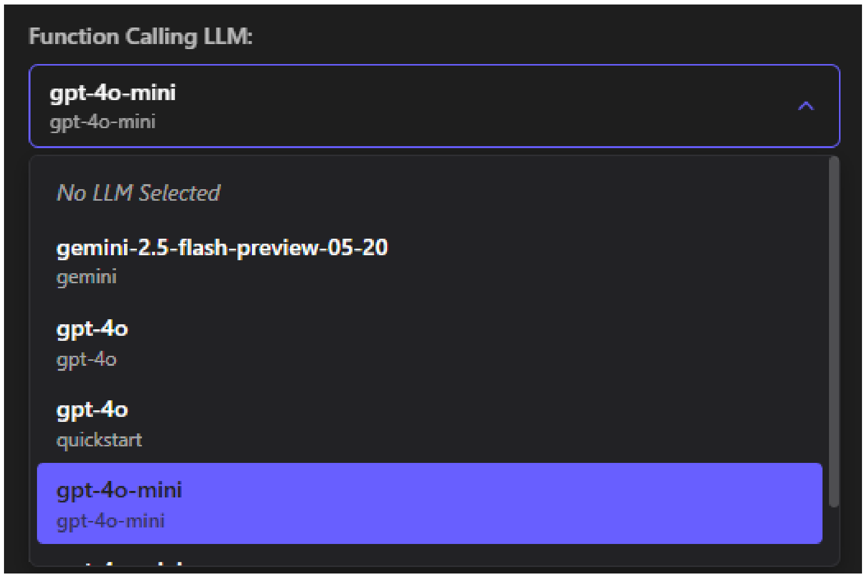


Advanced Settings for Hackathon Master

Agent Creativity Level: 0

Function Calling LLM

Select which language model should be used specifically for **function/tool calling**. Useful if you want to separate general reasoning from function execution.



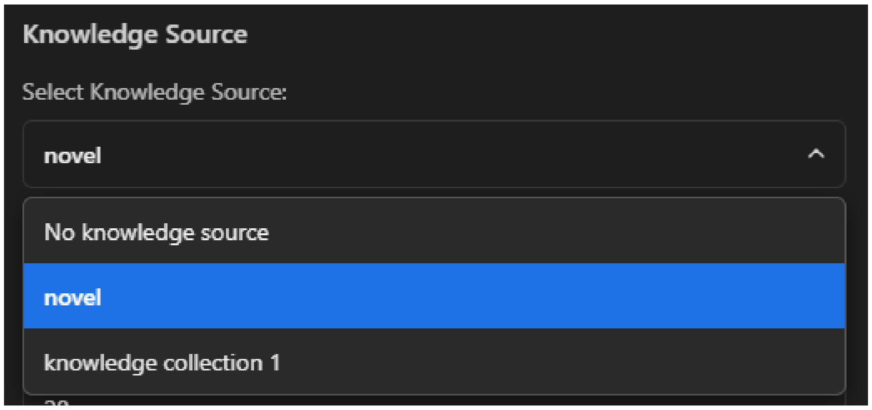
Function Calling LLM:

gpt-4o-mini
gpt-4o-mini

To find out more about setting up LLMs go here: [LLMs configurations]

Knowledge Source

Choose the data source the agent should rely on during execution — e.g., web search, internal docs, or no external knowledge.
Read how to create knowledge source here: [Knowledge]



Execution Settings

Fine-tune how the agent operates during a flow:

Maximum Iterations: Sets the maximum number of steps the agent can take to think, use tools, and generate results during one flow run.

Maximum Requests Per Minute: Throttles how often the agent can call tools or APIs.

Maximum Execution Time (seconds): Time limit for completing its tasks.

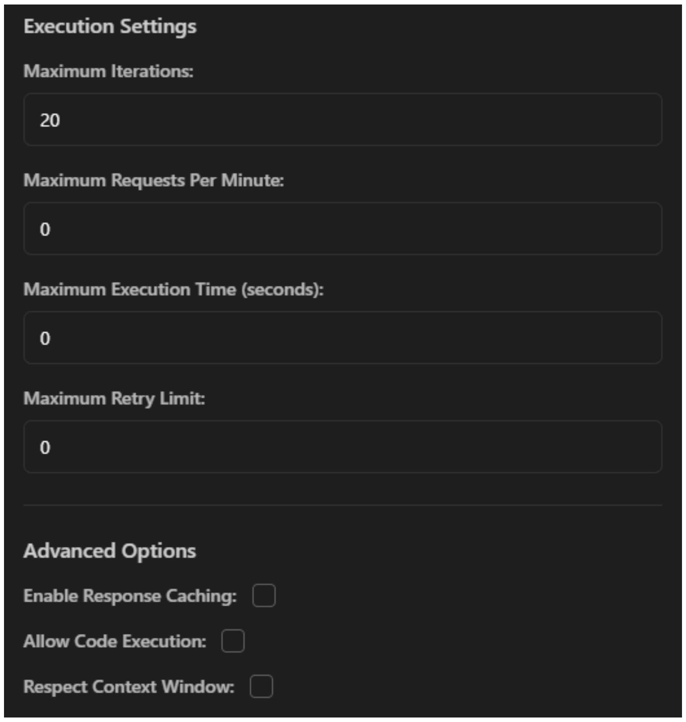
Maximum Retry Limit: How many times the agent will retry if something fails.

Advanced Options

Enable Response Caching: Caches the agent's responses to avoid repeating expensive operations.

Allow Code Execution: Allows the agent to run Python or other executable code when needed.

Respect Context Window: Forces the agent to stay within the context limit of its model (prevents overly long responses or memory overflows).


Execution Settings

Maximum Iterations: 20

Maximum Requests Per Minute: 0

Maximum Execution Time (seconds): 0

Maximum Retry Limit: 0

Advanced Options

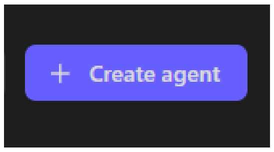
Enable Response Caching: ☐

Allow Code Execution: ☐

Respect Context Window: ☐

2. Creating an Agent via the “Create Agent” Modal

Another way to create an agent is by clicking the **“Create Agent”** button in the top right corner of the page. This opens a **modal window** with a form for entering the agent’s settings.

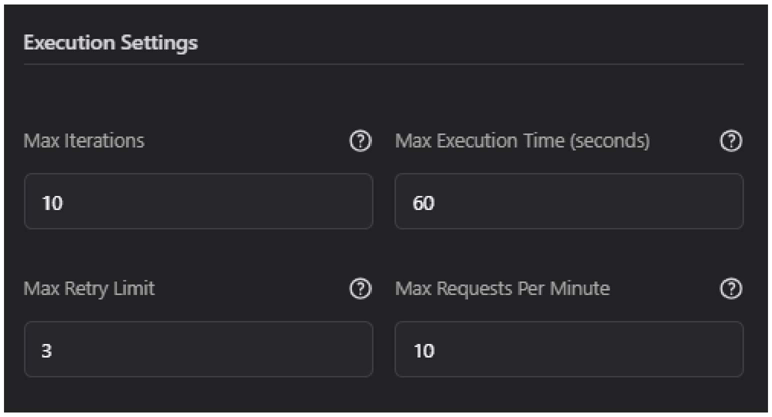


The form includes **almost the same fields** as in the table view — both the main properties and the advanced settings (normally hidden behind the gear icon).

Here are all the fields available in the modal:

- Agent Role
- Goal

- Backstory
- Agent LLM


Execution Settings

Max Iterations: 10

Max Execution Time (seconds): 60

Max Retry Limit: 3

Max Requests Per Minute: 10

⚠️Notes on Creating Agents

To successfully create an agent, the following fields are **required**:
Agent Role, **Goal**, and **Backstory**.

To ensure the agent can actually perform tasks in a flow, you also need to assign an **LLM model** (under **Agent LLM**).

All other fields are optional and can be customized later.

Table Actions (Right-Click Menu)

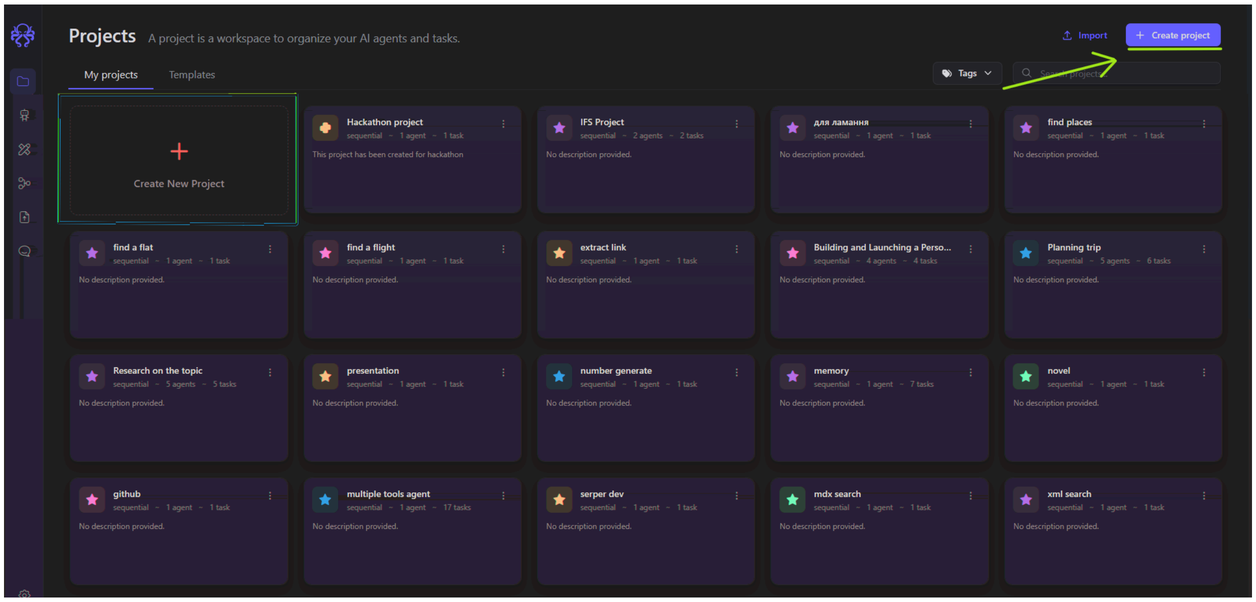
When working in the Table View, you can **right-click** on any row to access quick actions:

- **Add Empty Agent Above / Below**
  Inserts a new blank agent row directly above or below the selected one.
- **Copy Row**
  Copies the selected agent row (including all its settings).
- **Paste Row Above / Below**
  Pastes the previously copied agent into the table above or below the current one.
- **Delete Row**
  Removes the selected agent from the table.

project

Creating a Project

“Create Project”



Projects A project is a workspace to organize your AI agents and tasks.

Hackathon project
sequential ~ 1 agent ~ 1 task
This project has been created for hackathon

IFS Project
sequential ~ 2 agents ~ 2 tasks
No description provided.

для ламання
sequential ~ 1 agent ~ 1 task
No description provided.

find places
sequential ~ 1 agent ~ 1 task
No description provided.

find a flat
sequential ~ 1 agent ~ 1 task
No description provided.

find a flight
sequential ~ 1 agent ~ 1 task
No description provided.

extract link
sequential ~ 1 agent ~ 1 task
No description provided.

Building and Launching a Perso...
sequential ~ 4 agents ~ 4 tasks
No description provided.

Planning trip
sequential ~ 5 agents ~ 6 tasks
No description provided.

Research on the topic
sequential ~ 5 agents ~ 5 tasks
No description provided.

presentation
sequential ~ 1 agent ~ 1 task
No description provided.

number generate
sequential ~ 1 agent ~ 1 task
No description provided.

memory
sequential ~ 1 agent ~ 7 tasks
No description provided.

novel
sequential ~ 1 agent ~ 1 task
No description provided.

github
sequential ~ 1 agent ~ 1 task
No description provided.

multiple tools agent
sequential ~ 1 agent ~ 17 tasks
No description provided.

serper dev
sequential ~ 1 agent ~ 1 task
No description provided.

mdx search
sequential ~ 1 agent ~ 1 task
No description provided.

xml search
sequential ~ 1 agent ~ 1 task
No description provided.

modal window

Main Fields:

Project Name

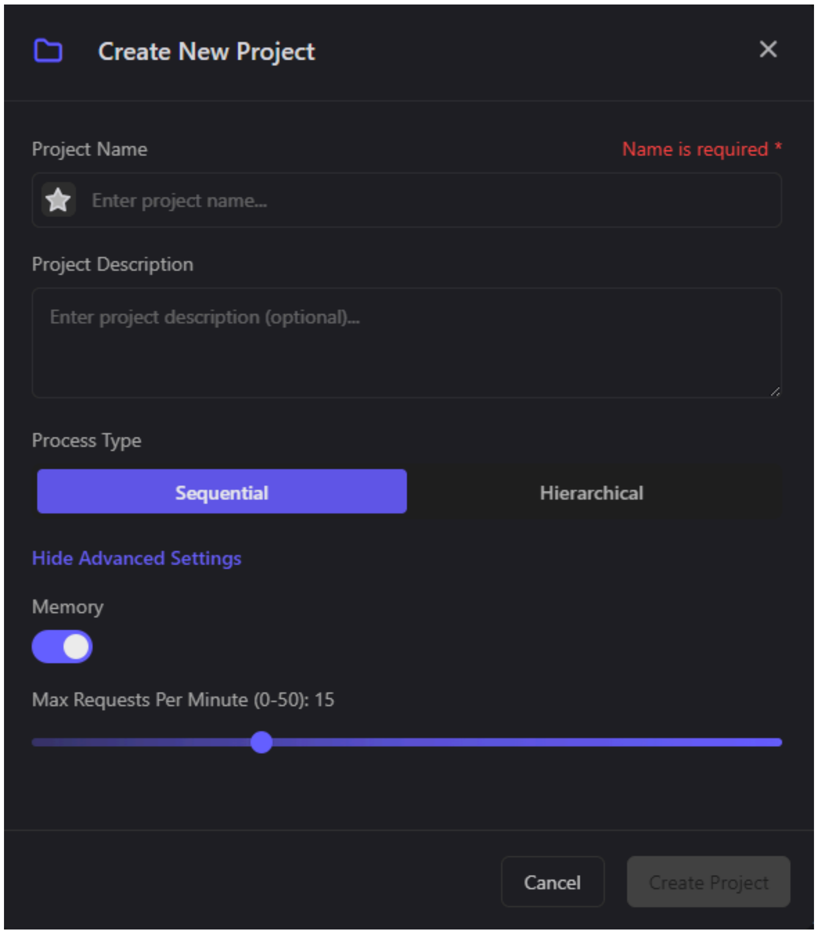
Project Description

Process Type

- **Sequential**
- **Hierarchical**

Advanced Settings:

Memory
Max Requests Per Minute


Create New Project

Project Name — Name is required *

Enter project name...

Project Description

Enter project description (optional)...

Process Type: [✓ Sequential] Hierarchical

Hide Advanced Settings

Memory: on

Max Requests Per Minute (0-50): 15

Cancel | Create Project

“Create”

Project Main Page

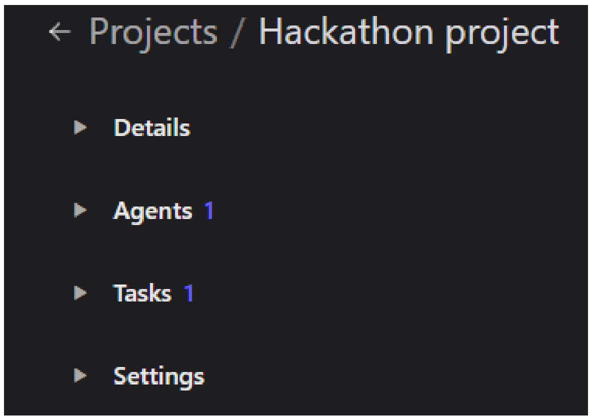
main page

Details

Agents

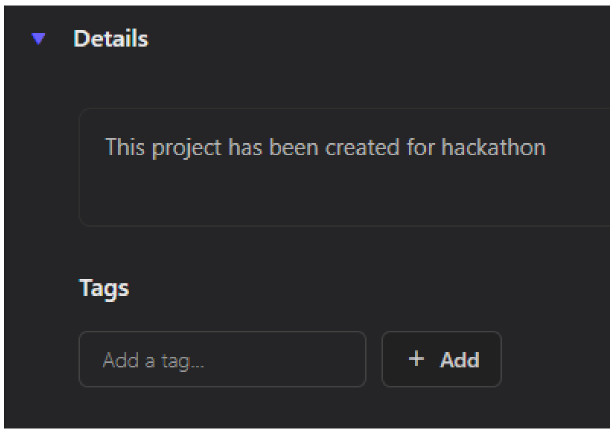
Tasks

Settings



1. Details

view or edit the project description



This project has been created for hackathon

2. Agents

“Search agents”

➕

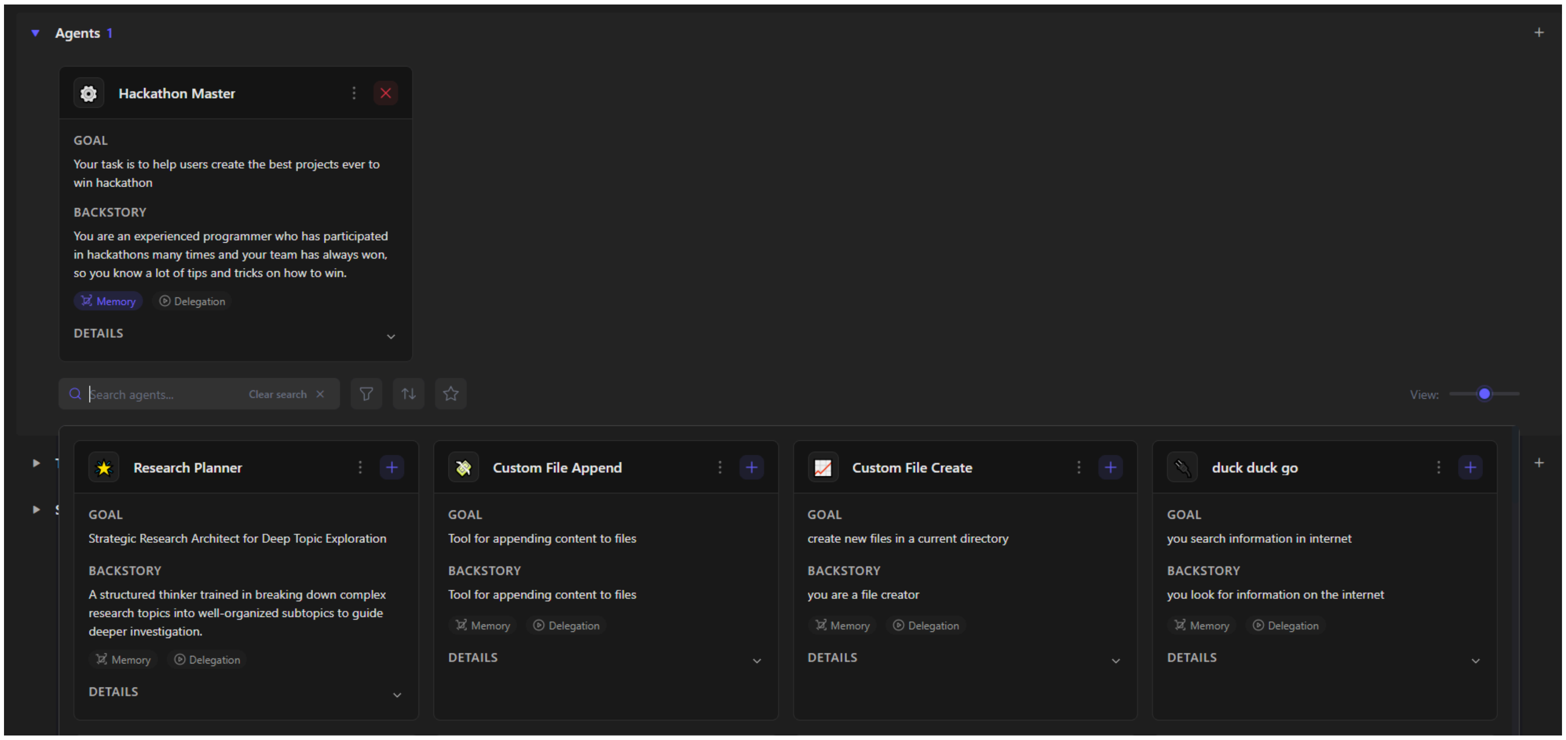
“Clear search”

“Details”

❌ cross

three dots Delete

Staff



Agents 1

Hackathon Master

GOAL

Your task is to help users create the best projects ever to win hackathon

BACKSTORY

You are an experienced programmer who has participated in hackathons many times and your team has always won, so you know a lot of tips and tricks on how to win.

Memory Delegation

DETAILS

Search agents... Clear search

View:

Research Planner

GOAL

Strategic Research Architect for Deep Topic Exploration

BACKSTORY

A structured thinker trained in breaking down complex research topics into well-organized subtopics to guide deeper investigation.

Memory Delegation

DETAILS

Custom File Append

GOAL

Tool for appending content to files

BACKSTORY

Tool for appending content to files

Memory Delegation

DETAILS

Custom File Create

GOAL

create new files in a current directory

BACKSTORY

you are a file creator

Memory Delegation

DETAILS

duck duck go

GOAL

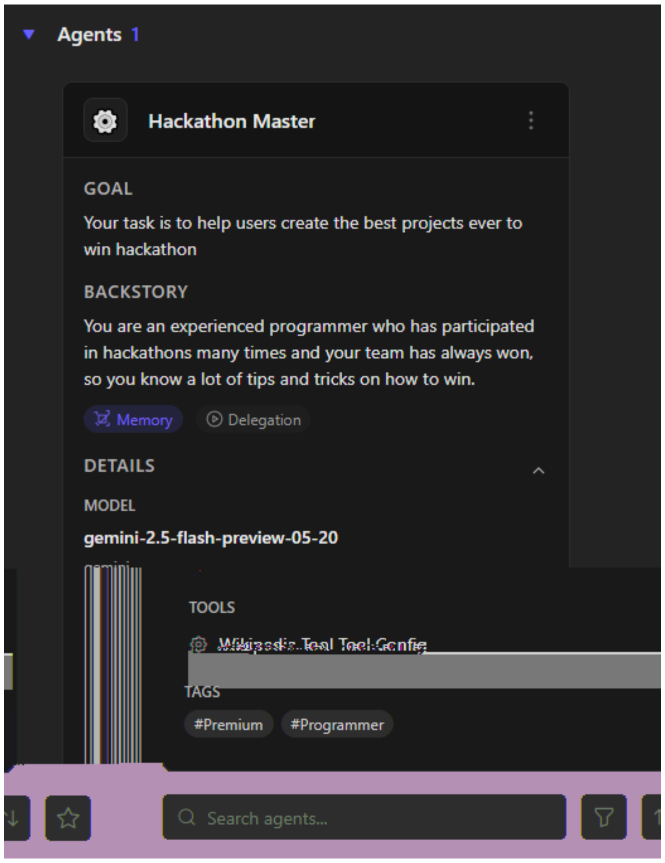
you search information in internet

BACKSTORY

you look for information on the internet

Memory Delegation

DETAILS



Hackathon Master

GOAL

Your task is to help users create the best projects ever to win hackathon

BACKSTORY

You are an experienced programmer who has participated in hackathons many times and your team has always won, so you know a lot of tips and tricks on how to win.

DETAILS

MODEL

gemini-2.5-flash-preview-05-20

TOOLS

Wikipedia Tool Tool Config

TAGS

#Premium #Programmer

3. Tasks

Required fields to define a task:

Task Name
Instructions

core prompt **task or action**

Good instructions are:

Examples:

Expected Output

define the format or structure

Examples:

[redacted]

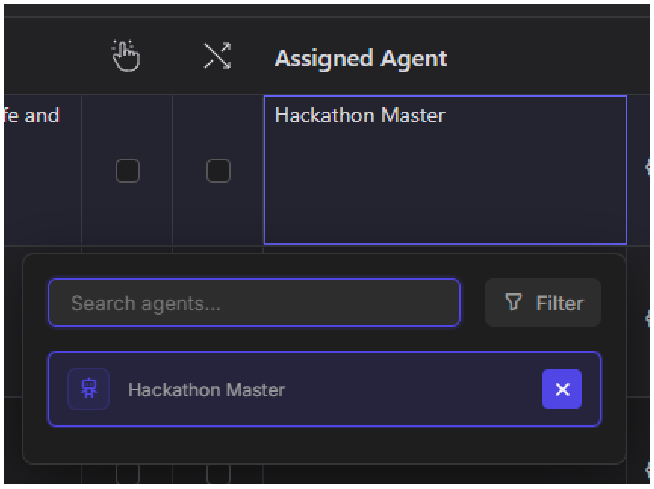
⚠️ **Important:**

must be assigned to an agent
cannot run without an agent

Assigning an agent to a task:

agent cell

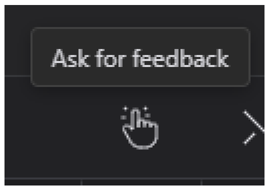
Variables



Additional options per task:

Human Input

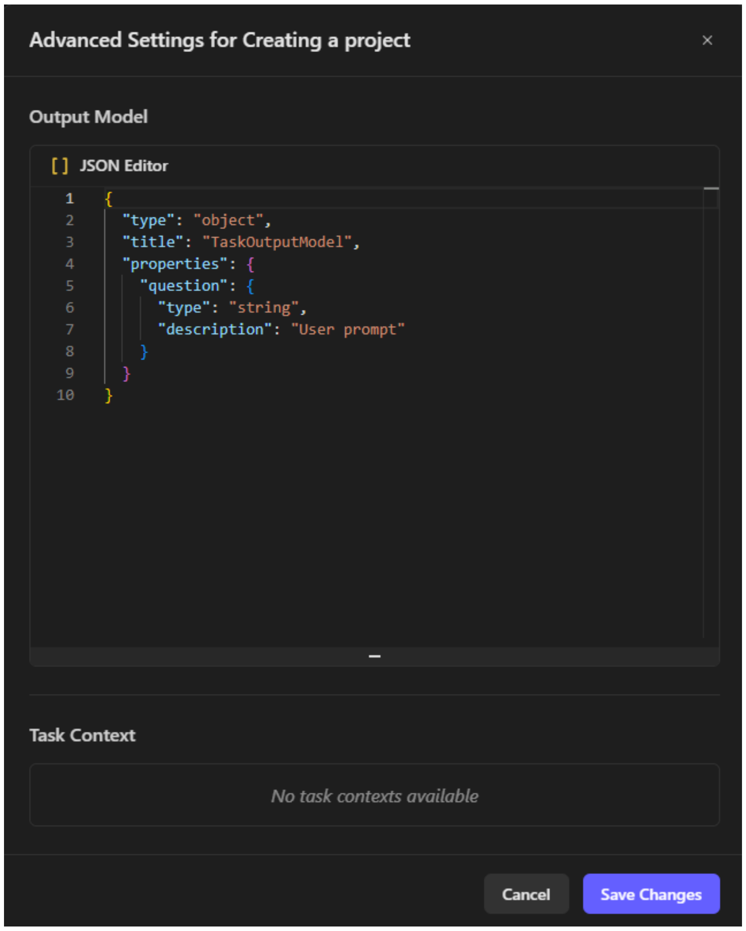
[Human Input]



Advanced task settings:

gear icon **Advanced Settings**
Output Model

question **description**
type **string**


Advanced Settings for Creating a project
×
Output Model
[ ] JSON Editor

```
{
  "type": "object",
  "title": "TaskOutputModel",
  "properties": {
    "question": {
      "type": "string",
      "description": "User prompt"
    }
  }
}
```

Task Context
No task contexts available
Cancel
Save Changes

Note

last task **Output Model** **project node**

4. Settings

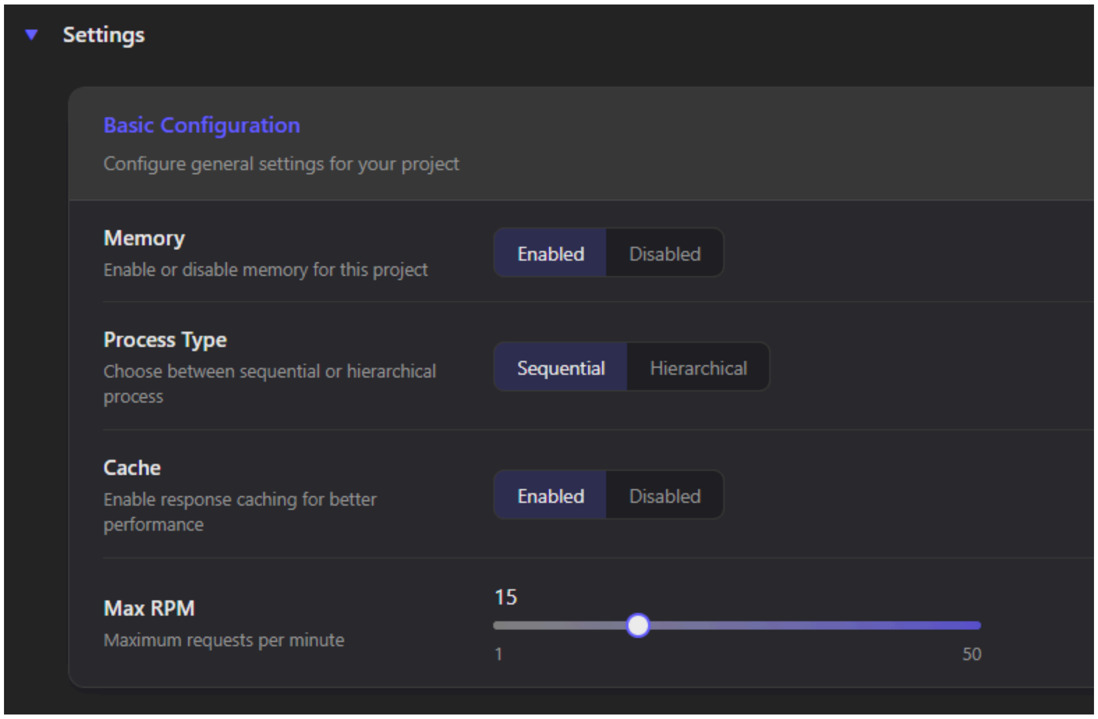
Memory

[Memory](#)

Process Type **Sequential** **Hierarchical**

Cache

Max Requests Per Minute (1–50)



Settings

Basic Configuration

Configure general settings for your project

Memory
Enable or disable memory for this project
[✓ Enabled] Disabled

Process Type
Choose between sequential or hierarchical process
[✓ Sequential] Hierarchical

Cache
Enable response caching for better performance
[✓ Enabled] Disabled

Max RPM
Maximum requests per minute
15
1 50

What are tools — and why do agents need them?

Tools a a c a a ca ca d c d b a a (LLM) ca d . T a **plugins**

[illegible]

a a :

a

ba d a da a

a - da a

c d c

R a

[illegible]

system

:

1. **lt-in tools** a d (. ., G S a c , Ca c a , W b Sc a)

d a a (. .,

[illegible]

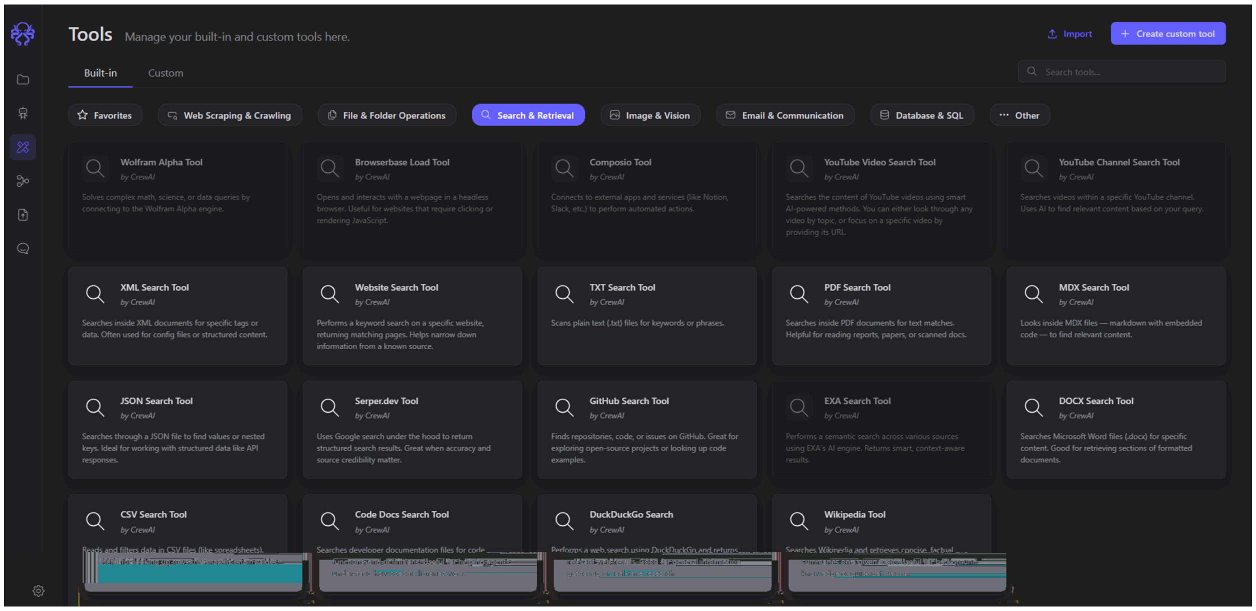
d a . T d b

T P c O ca a d b a ac d

ab

T

[illegible]


Tools Manage your built-in and custom tools here.

Import | Create custom tool

Built-in | Custom

Search tools...

Favorites | Web Scraping & Crawling | File & Folder Operations | Search & Retrieval | Image & Vision | Email & Communication | Database & SQL | Other

Wolfram Alpha Tool
by CrewAI
Solves complex math, science, or data queries by connecting to the Wolfram Alpha engine.

Browserbase Load Tool
by CrewAI
Opens and interacts with a webpage in a headless browser. Useful for websites that require clicking or rendering JavaScript.

Composio Tool
by CrewAI
Connects to external apps and services (like Notion, Slack, etc.) to perform automated actions.

YouTube Video Search Tool
by CrewAI
Searches the content of YouTube videos using smart AI-powered methods. You can either look through any video by topic, or focus on a specific video by providing its URL.

YouTube Channel Search Tool
by CrewAI
Searches videos within a specific YouTube channel. Uses AI to find relevant content based on your query.

XML Search Tool
by CrewAI
Searches inside XML documents for specific tags or data. Often used for config files or structured content.

Website Search Tool
by CrewAI
Performs a keyword search on a specific website, returning matching pages. Helps narrow down information from a known source.

TXT Search Tool
by CrewAI
Scans plain text (.txt) files for keywords or phrases.

PDF Search Tool
by CrewAI
Searches inside PDF documents for text matches. Helpful for reading reports, papers, or scanned docs.

MDX Search Tool
by CrewAI
Looks inside MDX files — markdown with embedded code — to find relevant content.

JSON Search Tool
by CrewAI
Searches through a JSON file to find values or nested keys. Ideal for working with structured data like API responses.

Serper.dev Tool
by CrewAI
Uses Google search under the hood to return structured search results. Great when accuracy and source credibility matter.

GitHub Search Tool
by CrewAI
Finds repositories, code, or issues on GitHub. Great for exploring open-source projects or looking up code examples.

EXA Search Tool
by CrewAI
Performs a semantic search across various sources using EXA's AI engine. Returns smart, context-aware results.

DOCX Search Tool
by CrewAI
Searches Microsoft Word files (.docx) for specific content. Good for retrieving sections of formatted documents.

CSV Search Tool
by CrewAI
Reads and filters data in CSV files (like spreadsheets).

Code Docs Search Tool
by CrewAI
Searches developer documentation files for code ...

DuckDuckGo Search
by CrewAI
Performs a web search using DuckDuckGo and returns...

Wikipedia Tool
by CrewAI
Searches Wikipedia and retrieves concise, factual ...

Tool Configurations

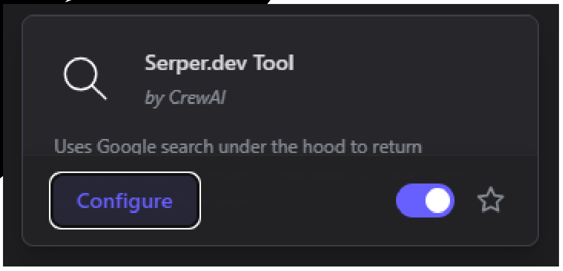
S add a c a .

[redacted]

...gure ... n tool:

. a a a .

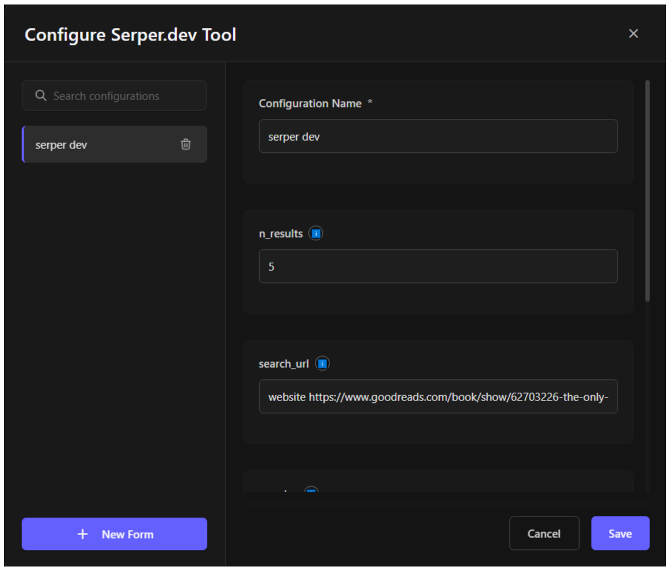
[redacted]


Serper.dev Tool
by CrewAI
Uses Google search under the hood to return

The configuration form includes:

➢ **A name for the configuration (** d a)
➢ **Additional parameters (** a h)

[redacted]


Configure Serper.dev Tool

Search configurations

serper dev

Configuration Name *

serper dev

n_results

5

search_url

website https://www.goodreads.com/book/show/62703226-the-only-

New Form

Cancel

Save

⚠️F d [illegible] a **blue info icon** a [illegible] ad a [illegible] a a [illegible] .

⚠️ [illegible] sk [illegible] ab [illegible] **Create** [illegible]

1. C c [illegible] **New** [illegible]
2. F a a [illegible] d d [illegible] a d [illegible] *).
3. C c **Creat** [illegible]

[illegible]

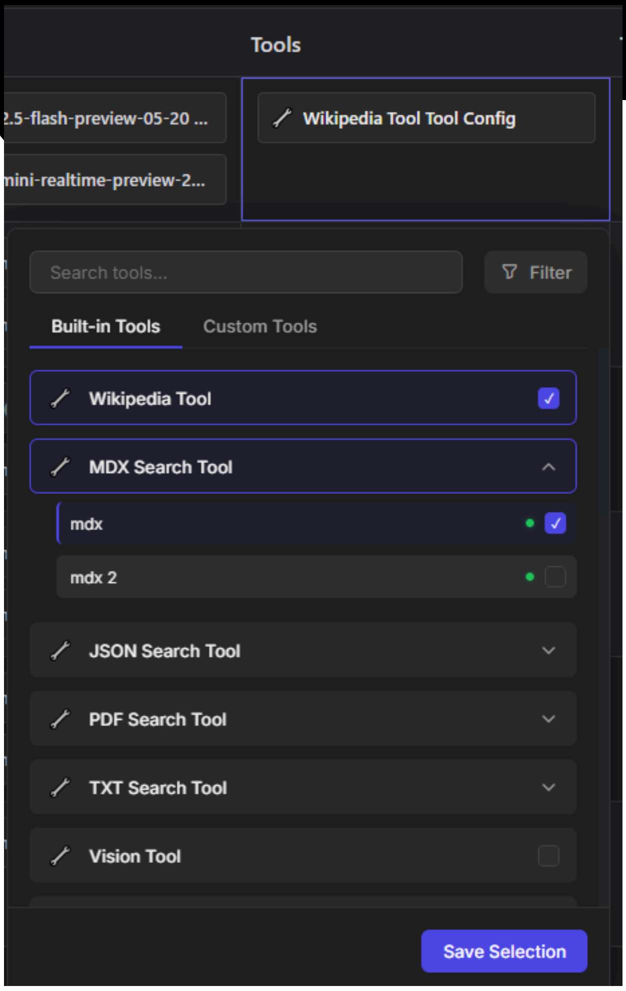
1. [illegible] c [illegible] c a [illegible] .
2. [illegible] Ma [illegible] a [illegible] .

[illegible] a built-in tool to an agent

1. G [illegible] **Agents** c [illegible] c .
2. D [illegible] -c c [illegible] c d [illegible] **T**[illegible]**s** c [illegible] d d a [illegible] . [illegible] d [illegible] **“Built-in”** [illegible]

[illegible] **own menu** a a .

6. S c c a b a a d b a d a a d a [redacted]



Tools

2.5-flash-preview-05-20 ...

mini-realtime-preview-2...

Wikipedia Tool Tool Config

Search tools...

Filter

Built-in Tools

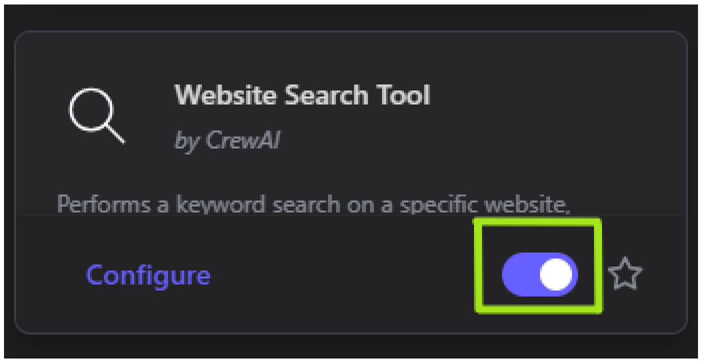
Custom Tools

- Wikipedia Tool ✓
- MDX Search Tool
  - mdx ● ✓
  - mdx 2 ●
- JSON Search Tool
- PDF Search Tool
- TXT Search Tool
- Vision Tool

Save Selection

Disabling Tools

Y ca a **disable** a b c a a a [redacted]



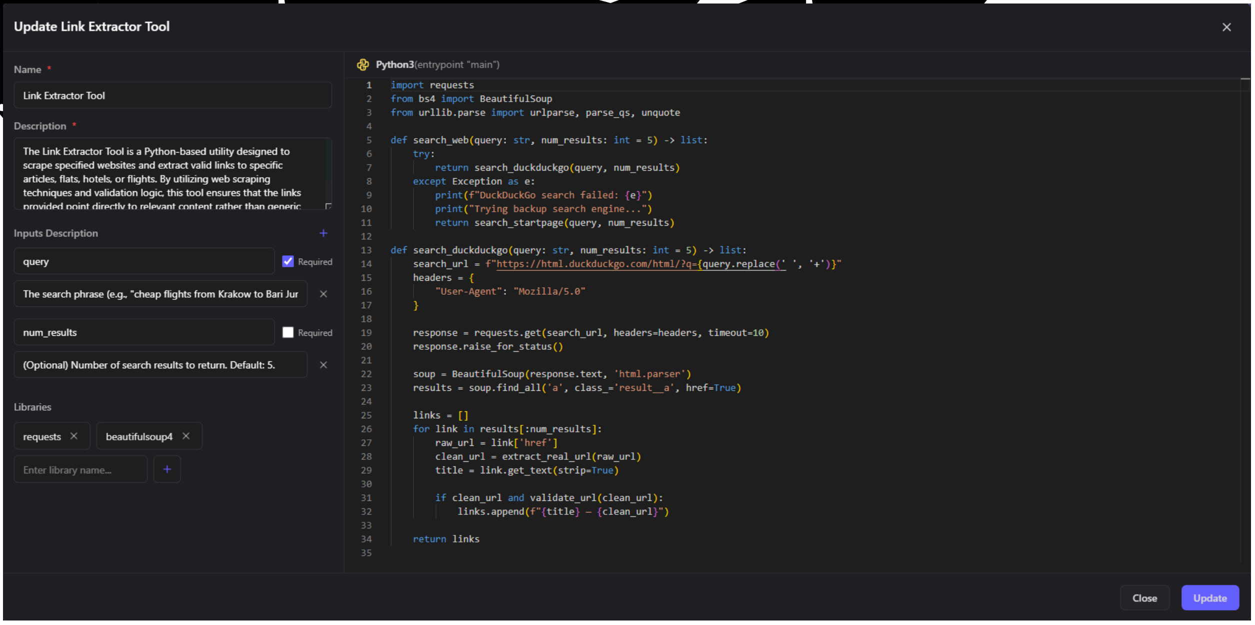
Website Search Tool

by CrewAI

Performs a keyword search on a specific website.

🐍 Custom Tools (Python)

[redacted]



Update Link Extractor Tool

Name *

Link Extractor Tool

Description *

The Link Extractor Tool is a Python-based utility designed to scrape specified websites and extract valid links to specific articles, flats, hotels, or flights. By utilizing web scraping techniques and validation logic, this tool ensures that the links provided point directly to relevant content rather than generic

Inputs Description

query [✓ Required]

The search phrase (e.g., "cheap flights from Krakow to Bari Jur

num_results [ ] Required

(Optional) Number of search results to return. Default: 5.

Libraries

requests, beautifulsoup4

Enter library name...

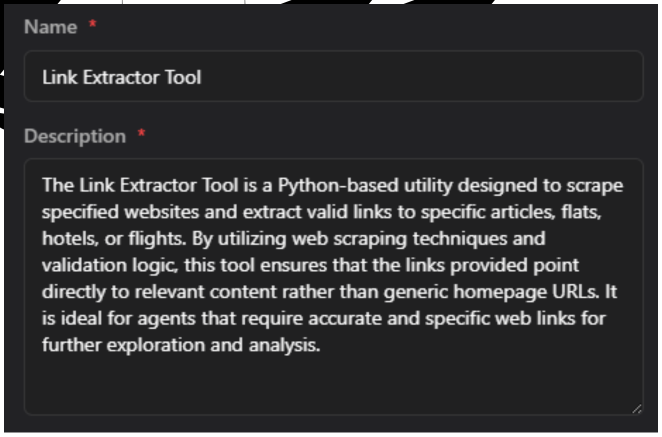
Python3(entrypoint "main")

```python
import requests
from bs4 import BeautifulSoup
from urllib.parse import urlparse, parse_qs, unquote

def search_web(query: str, num_results: int = 5) -> list:
    try:
        return search_duckduckgo(query, num_results)
    except Exception as e:
        print(f"DuckDuckGo search failed: {e}")
        print("Trying backup search engine...")
        return search_startpage(query, num_results)

def search_duckduckgo(query: str, num_results: int = 5) -> list:
    search_url = f"https://html.duckduckgo.com/html/?q={query.replace(' ', '+')}"
    headers = {
        "User-Agent": "Mozilla/5.0"
    }

    response = requests.get(search_url, headers=headers, timeout=10)
    response.raise_for_status()

    soup = BeautifulSoup(response.text, 'html.parser')
    results = soup.find_all('a', class_='result__a', href=True)

    links = []
    for link in results[:num_results]:
        raw_url = link['href']
        clean_url = extract_real_url(raw_url)
        title = link.get_text(strip=True)

        if clean_url and validate_url(clean_url):
            links.append(f"{title} – {clean_url}")

    return links

```

Required Fields:

Name [redacted] a

Descripti [redacted] a a [redacted] a [redacted] d



Name *

Link Extractor Tool

Description *

The Link Extractor Tool is a Python-based utility designed to scrape specified websites and extract valid links to specific articles, flats, hotels, or flights. By utilizing web scraping techniques and validation logic, this tool ensures that the links provided point directly to relevant content rather than generic homepage URLs. It is ideal for agents that require accurate and specific web links for further exploration and analysis.

Inputs Description:

C c [redacted] + c [redacted] add [redacted] a a [redacted] `main()` [redacted] c

E [redacted] ac [redacted] , [redacted] d :

[redacted] a a [redacted] a [redacted] P [redacted] c d .

[redacted] a [redacted] c

`(main()` [redacted] a a [redacted] c

[redacted]

`def main(city: str, date: str) -> dict:`

Y [redacted] d [redacted] a [redacted] :

- `city`
- `date`

[redacted] a c [redacted] ly [redacted] ca - [redacted] , [redacted] .

Descripti [redacted]

[redacted] a [redacted] d

[redacted]

- F city: *"Destination city for flight search"*
- F date: *"Preferred departure date in YYYY-MM-DD format"*

W ' : T d c a a (c □),
[illegible]

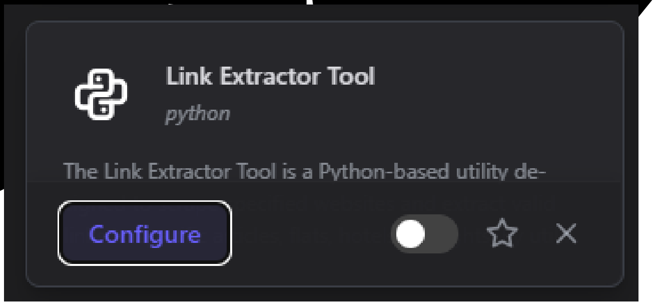
Libraries:

I c d d a b a , **“Enter library name…”**
[illegible]

⚠️ [illegible] +

⚠️ [illegible] *raries manually — some are*
[illegible]

A c a d a a **Custom Tools** ab T a
[redacted] a d c c **Configure**
[redacted] **× (cross)** [redacted]



Link Extractor Tool

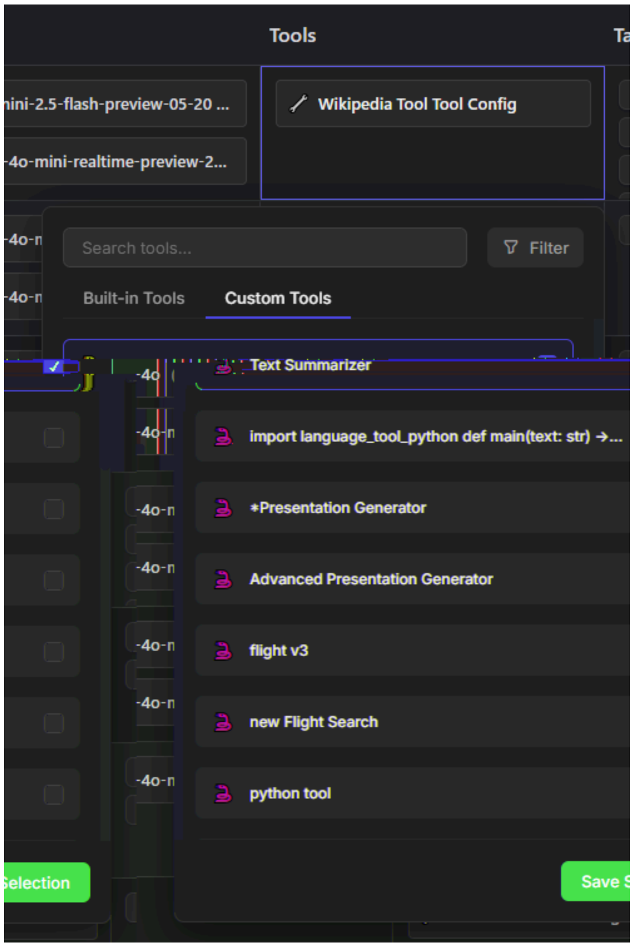
python

The Link Extractor Tool is a Python-based utility de-

Assigning a Custom Tool to an Agent

1. G [redacted] **Staff Table**
2. D [redacted] -c c c d **Tools** c
[redacted] **Custom Tools** ab
a

[redacted]

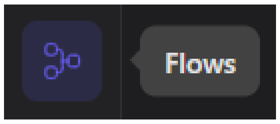

Tools
Ta
nini-2.5-flash-preview-05-20 ...
-4o-mini-realtime-preview-2...
Wikipedia Tool Tool Config
-4o-n
-4o-n
Search tools...
Filter
Built-in Tools
Custom Tools
Text Summarizer
import language_tool_python def main(text: str) →...
*Presentation Generator
Advanced Presentation Generator
flight v3
new Flight Search
python tool
-4o
-4o-n
election
Save S

R ad c a a d c a : Staff

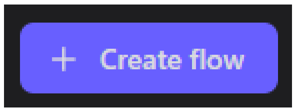
Flows let you automate how agents and tools interact in your project. Think of it as a visual pipeline where you define *what happens, in what order, and under what conditions*. Every project must be launched through a flow.

Step 1: Create a New Flow

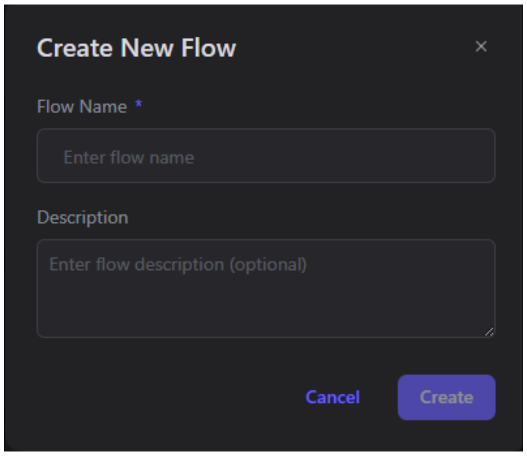
1. Go to the **Flows** page



2. Click the ➕ **Create Flow** button (top right corner)



3. A small window will open – enter:
    - **Flow Name** (required)
    - **Description** (optional)



Create New Flow ×

Flow Name *

Enter flow name

Description

Enter flow description (optional)

Cancel Create

4. Click **Create**

You'll now be taken to the flow's canvas – a blank space where you'll build the logic of your flow.

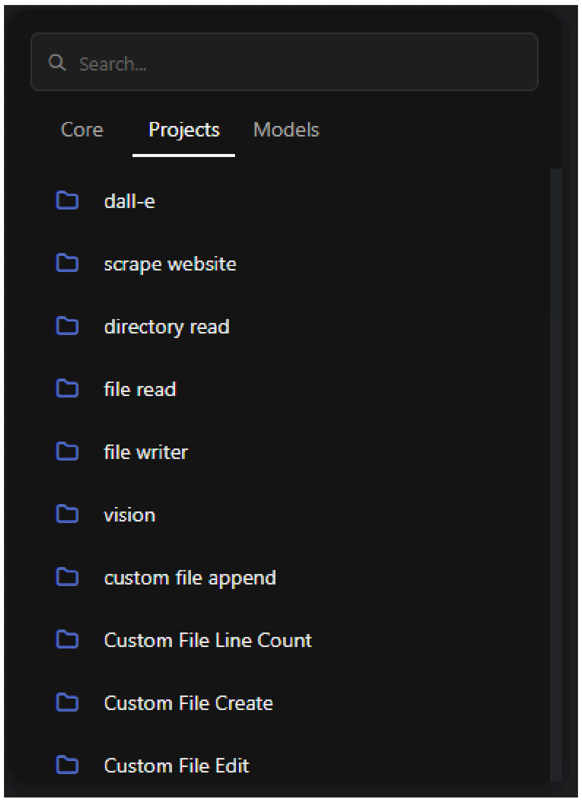
Step 2: Understand the flow canvas

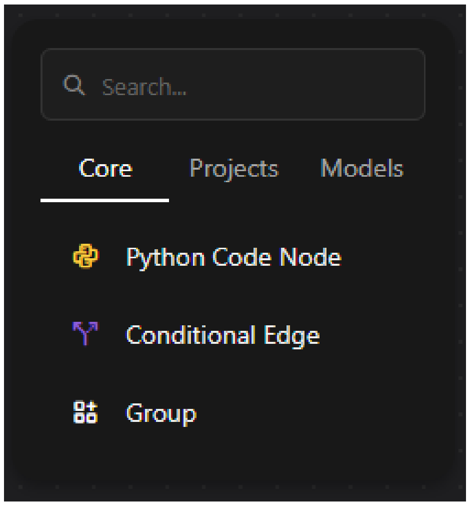
Each flow starts with a **Start Node** – it's added automatically and is always the entry point.

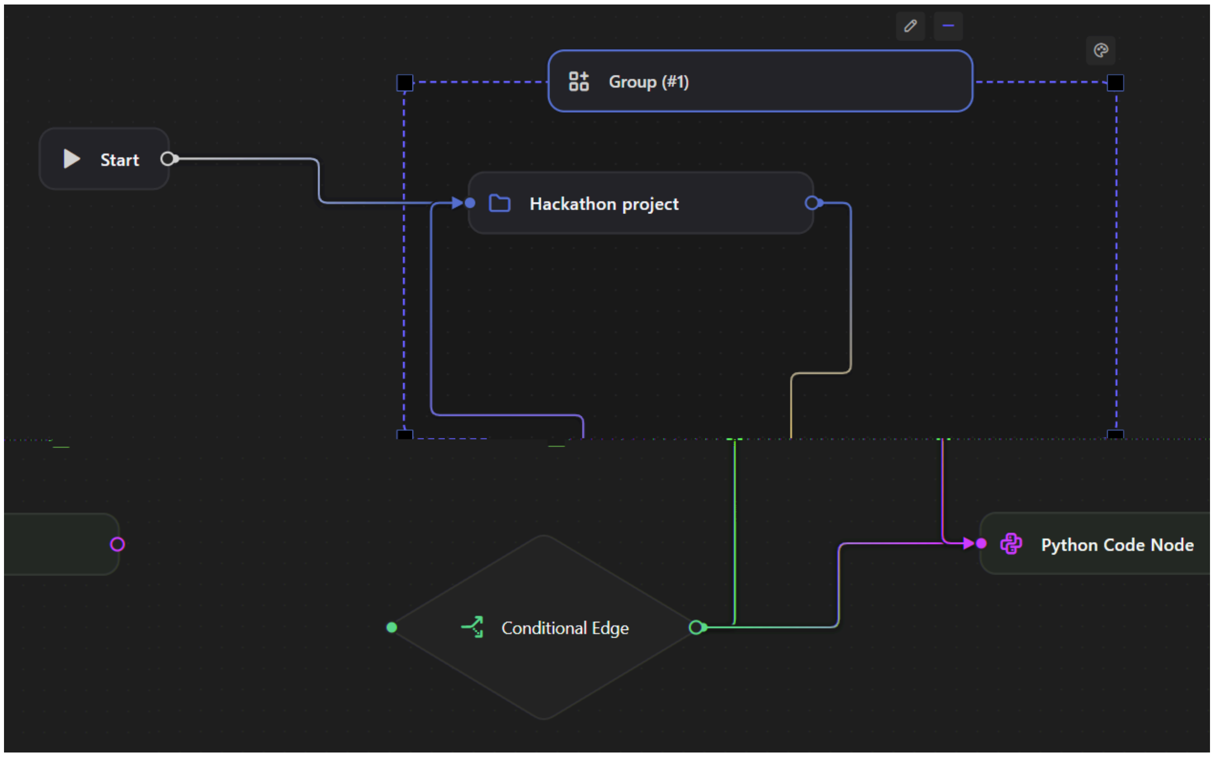
From there, you can add and connect different types of nodes:

- **Python Node** – run Python code
- **Project Node** – run previously created projects
- **Conditional Edge** – set conditions (like if/else branches)
- **Group** – organize your nodes visually

You can find out more about nodes and how to manage them here: Nodes







Step 3: Add nodes to the canvas

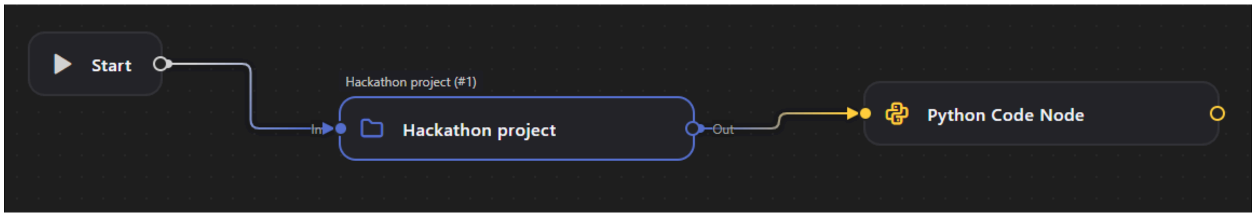
1. **Right-click** anywhere on the canvas
2. A popup will appear with node types
3. Navigate the tabs and click the type you want to add

The node will be placed on the canvas. You can **drag to reposition it** as needed.

For better navigation you can click on the magnifying glass icon, then you will see the list of all nodes present on your canvas. Click on any node and you will be redirected to it.

Step 4: Connect the nodes (Define the Flow)

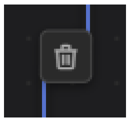
1. **Click and drag** from one node's output port to another node's input
2. Arrows will appear between them
3. This defines the **execution order** – from Start to Finish



⚠️ ***Important***: Your flow will run *exactly* in the order of these connections.

To **delete a connection**:

Click on the arrow and press **Delete** or use the **trash icon**



To delete a node: click it, then use the delete key or the trash icon.

Step 5: Configure each node

Each node has **its own settings**:

- Click on a node to open its configuration panel
- You can:
    - Set **variables** (e.g., pass data from one node to another)
    - Rename the node (each node must have a **unique name**)

We explain **variables** separately – but they let nodes exchange data and results. Here's the page dedicated to variables

---

Step 6: Save and run the Flow

Before running, you **must save** your flow:

1. Click **Save** (top right corner)

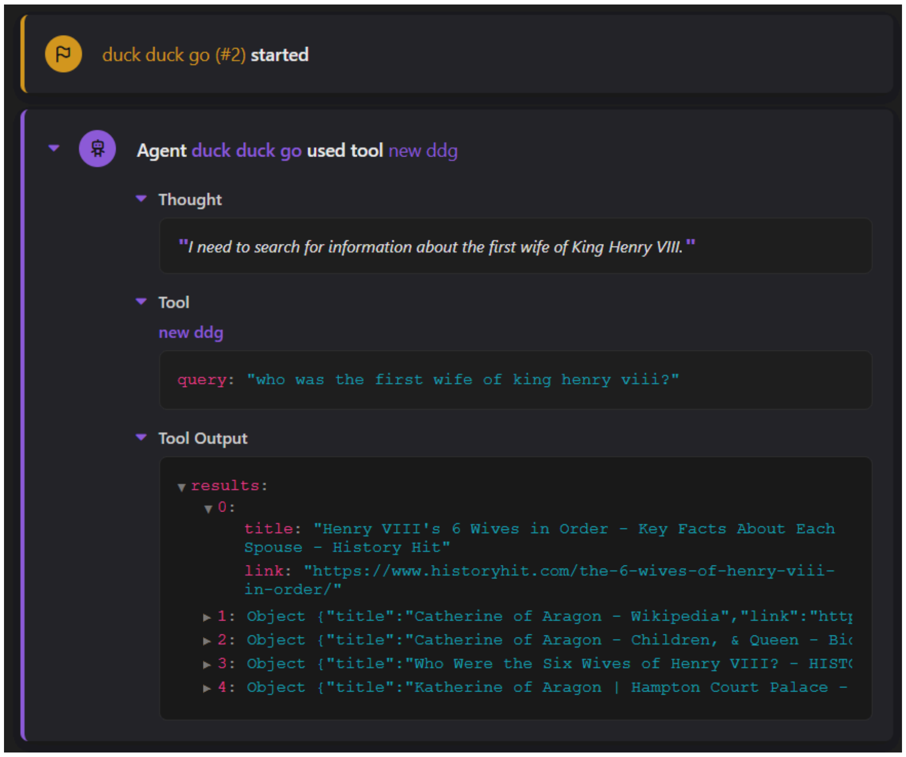
2. Then click **Run**

You will be redirected to the **Run Session Page**, where you can watch your flow execute live.

Step 7: What happens during execution?

As your flow runs:

- You'll see nodes activate one by one, **in the defined order**
- For each **Project Node** (agent task):
    - You'll see:What the agent was asked to do
        - Their thought process (reasoning)
        - If a tool was used, you'll see:
            - The **query**
            - The **tool output**
        - Final result or decision



duck duck go (#2) started

Agent duck duck go used tool new ddg

Thought

"I need to search for information about the first wife of King Henry VIII."

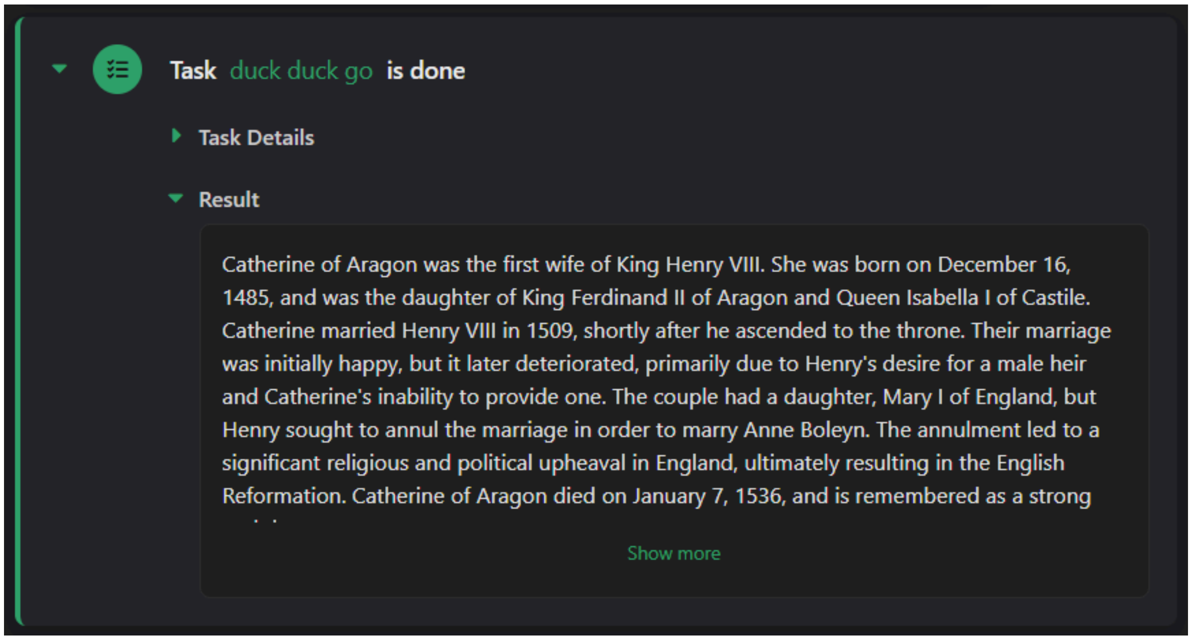
Tool

new ddg

```
query: "who was the first wife of king henry viii?"
```

Tool Output

```
▾results:
  ▾0:
     title: "Henry VIII's 6 Wives in Order - Key Facts About Each
     Spouse - History Hit"
     link: "https://www.historyhit.com/the-6-wives-of-henry-viii-
     in-order/"
  ▸1: Object {"title":"Catherine of Aragon - Wikipedia","link":"htt
  ▸2: Object {"title":"Catherine of Aragon - Children, & Queen - Bi
  ▸3: Object {"title":"Who Were the Six Wives of Henry VIII? - HIST
  ▸4: Object {"title":"Katherine of Aragon | Hampton Court Palace -
```



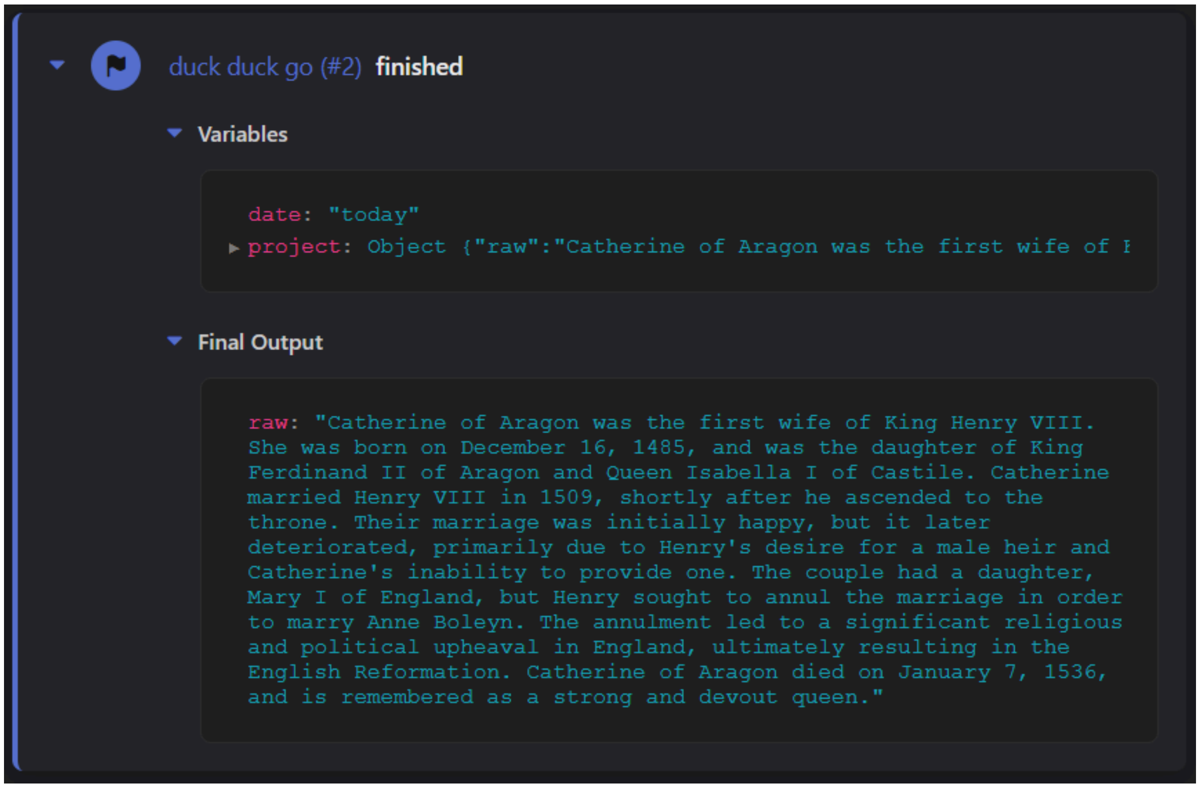
Task duck duck go is done

Task Details

Result

Catherine of Aragon was the first wife of King Henry VIII. She was born on December 16, 1485, and was the daughter of King Ferdinand II of Aragon and Queen Isabella I of Castile. Catherine married Henry VIII in 1509, shortly after he ascended to the throne. Their marriage was initially happy, but it later deteriorated, primarily due to Henry's desire for a male heir and Catherine's inability to provide one. The couple had a daughter, Mary I of England, but Henry sought to annul the marriage in order to marry Anne Boleyn. The annulment led to a significant religious and political upheaval in England, ultimately resulting in the English Reformation. Catherine of Aragon died on January 7, 1536, and is remembered as a strong

Show more

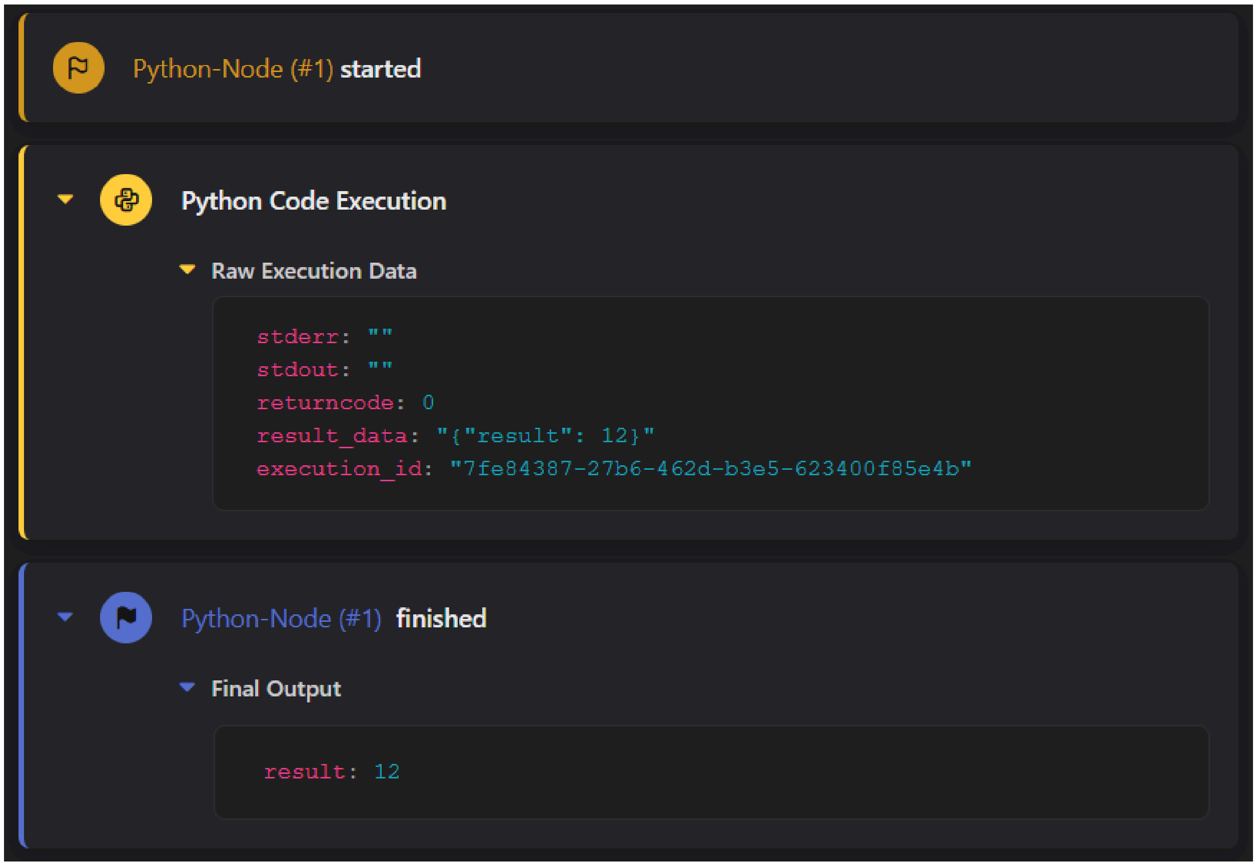

duck duck go (#2) finished

Variables

```
date: "today"
▸ project: Object {"raw":"Catherine of Aragon was the first wife of I
```

Final Output

```
raw: "Catherine of Aragon was the first wife of King Henry VIII.
She was born on December 16, 1485, and was the daughter of King
Ferdinand II of Aragon and Queen Isabella I of Castile. Catherine
married Henry VIII in 1509, shortly after he ascended to the
throne. Their marriage was initially happy, but it later
deteriorated, primarily due to Henry's desire for a male heir and
Catherine's inability to provide one. The couple had a daughter,
Mary I of England, but Henry sought to annul the marriage in order
to marry Anne Boleyn. The annulment led to a significant religious
and political upheaval in England, ultimately resulting in the
English Reformation. Catherine of Aragon died on January 7, 1536,
and is remembered as a strong and devout queen."
```


Python-Node (#1) started

Python Code Execution

Raw Execution Data

```
stderr: ""
stdout: ""
returncode: 0
result_data: "{"result": 12}"
execution_id: "7fe84387-27b6-462d-b3e5-623400f85e4b"
```

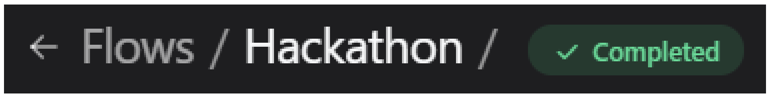
Python-Node (#1) finished

Final Output

```
result: 12
```

Execution Statuses – what do they mean?

At the top of the run screen, you'll see the **flow status**:



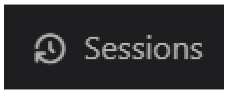
- **Completed** – Flow finished successfully
- **Running** – Flow is actively executing
- **Pending** – Flow is waiting to start
- **Waiting for Human Input** – Agent needs a user reply
- **Error** – Something went wrong
- **Expired** – Flow ran too long or was interrupted

Step 8: Reviewing sessions (past flow runs)

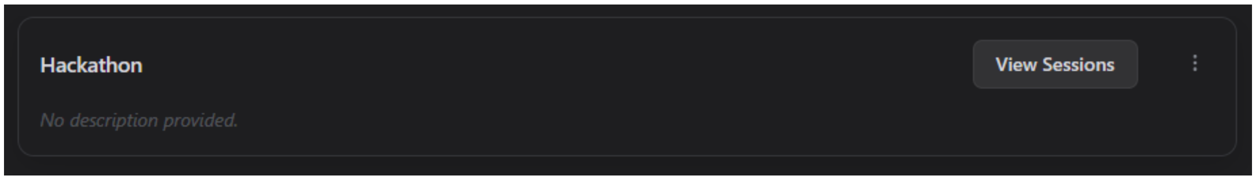
Each time you run a flow, it creates a **session** – a full log of what happened.

You can find past sessions in two places:

1. On the Run Page: click the **Sessions** button (top right)

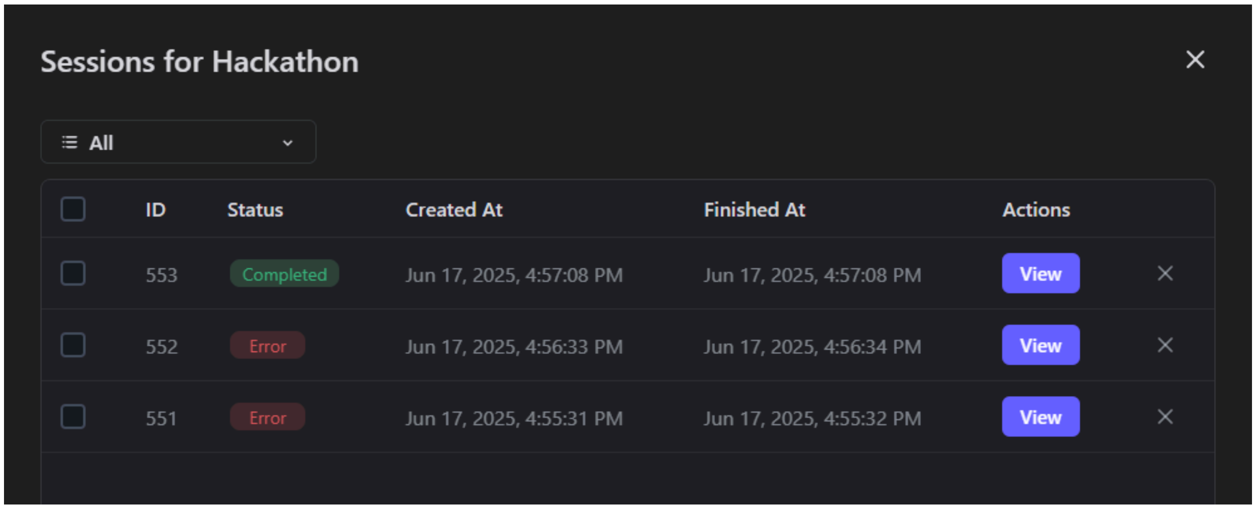


2. On the Flows list: click **View Sessions** next to any flow



From the sessions panel, you can:

- **Open** any session to review it step-by-step

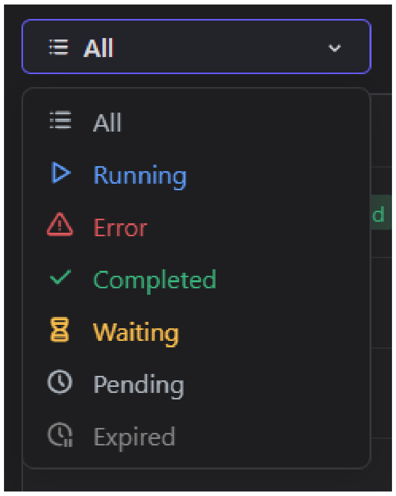


Sessions for Hackathon

All

ID	Status	Created At	Finished At	Actions
553	Completed	Jun 17, 2025, 4:57:08 PM	Jun 17, 2025, 4:57:08 PM	View ✕
552	Error	Jun 17, 2025, 4:56:33 PM	Jun 17, 2025, 4:56:34 PM	View ✕
551	Error	Jun 17, 2025, 4:55:31 PM	Jun 17, 2025, 4:55:32 PM	View ✕

- **Filter** by status



- **Delete** sessions using the ❌ icon

Important Notes

- You can't run a project *without* a flow
- You **must save** before running – otherwise your latest changes won't be applied
- Use clear and unique names for nodes to stay organized

Node Types in Flows

Each flow is made up of **nodes**, which are the building blocks of automation. Here's a breakdown of every node type, what it does, and how it can be configured.

1. Start Node

- This node is always present by default in every flow.
- It **does not perform any action**, but acts as the entry point of your flow.
- You can pass initial **variables** into it using JSON — these variables will be available to all following nodes.

You can find a detailed explanation about variables here: Variables

---

2. Python Node

A Python Node lets you run custom Python code inside your flow. Use it when you need to:

- Transform or filter data
- Do calculations or logic
- Work with external APIs or data
- Prepare values for agents or later steps

Configuration Fields

When you add a Python Node, you'll see these fields:

Field	‘What it does
Name	The name of your Python tool (used in tool lists, not flow logic)
Node name	A unique name for this node inside the flow (helps you identify it)

Input map	Defines which flow variables are passed into your Python main() function
Output variable path	Where the result will be stored as a flow variable
Libraries	Python libraries your code needs. Press Enter/+ after typing each

You can find a -

JSON

Python

This will multiply the values of variables.price and variables.quantity, and return the result.

To save the result, you must fill in Output variable path:

```
None
```

➡ You can use variables.total_price in later nodes or agents.

Example 2 — Without input map:

If you don’t use any variables or inputs, your code could look like this:

```
Python
```

In this case, the node always returns 42 — not dynamic or reusable.

Output variable path

This tells the system where to store your result from the Python code.

If your main() returns:

```
Python
```

And your Output variable path is:

```
None
```

Then variables.my_result will contain the output and can be used by other nodes.

Summary

- Use Input map to pass values into main() from your flow.
- Your main() must return a dictionary like {"result": value}.
- Use Output variable path to save the result into a variable.
- If you don't use the input map, your code can still run — but values will be hardcoded.
- Always press Save after making changes to the node!

---

3. Project Node

A **Project Node** lets you run any **existing project** inside a flow. It triggers a project you've already built in the system.

Configuration Fields

When you add a Project Node, you'll see the following fields:

Field	What it does
Node name	A unique name for this node inside your flow
Input map	Pass values from flow variables into the project
Output variable path	Save the final result of the project into a flow variable

How does it work?

When the flow reaches a **Project Node**, it:

1. Loads the selected **project**
2. If provided, uses the values from the **Input map** to populate variables inside the project
3. Executes the project step-by-step
4. Can save the final **output** into the variable you define in **Output variable path**

Example

Let’s say your project expects a variable called city.

Input map:

JSON
```
{



}
```

If variables.destination_city is "Paris", then the project will run using city = "Paris".

If the result of the project is a recommendation list, and you write:

None
```

```

in the **Output variable path**, then the output will be saved and can be used in later nodes.

You can find a detailed explanation about variables here: Variables

---

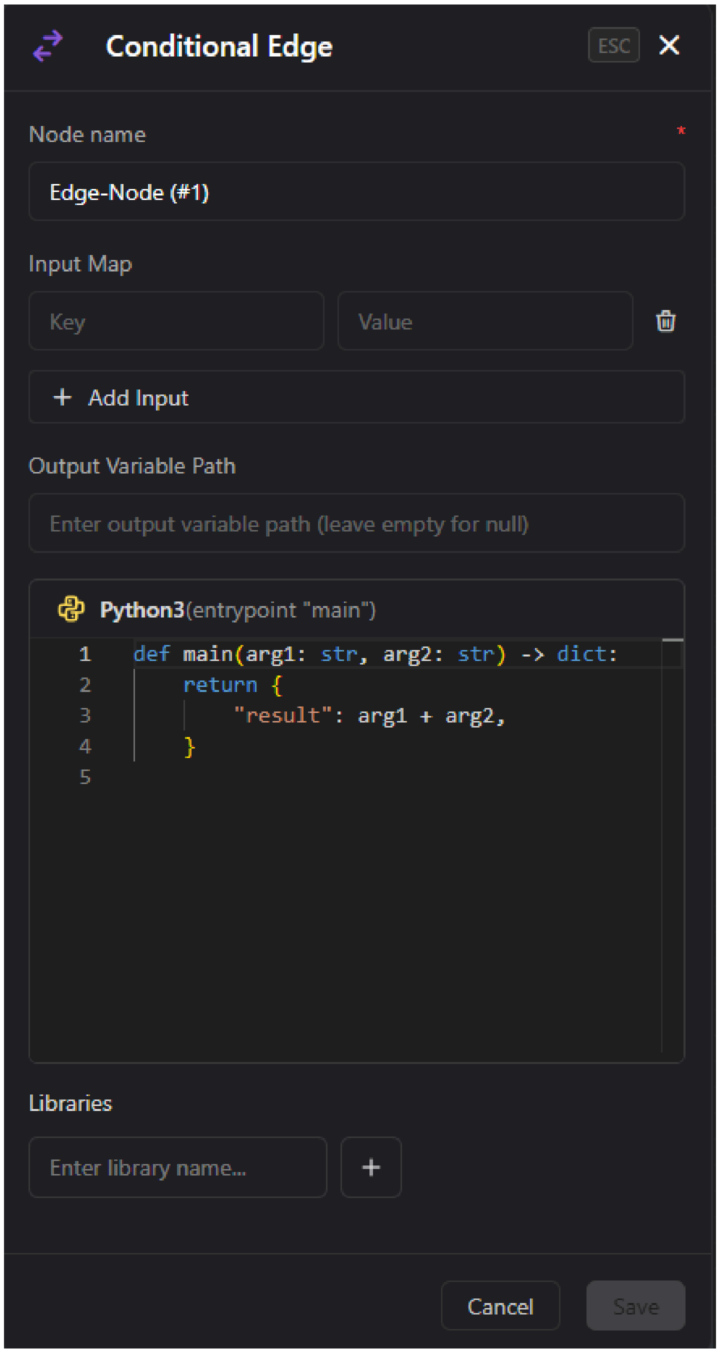
4. Conditional Edge

Fik

A **Conditional Edge** is a special type of node that helps your flow decide **which path to follow next**, based on logic written in Python.

It’s like an “if/else” statement for your flow.

Configuration Fields



```python
def main(arg1: str, arg2: str) -> dict:
    return {
        "result": arg1 + arg2,
    }
```

How does it work?

1. In the **Python code box**, you write logic like:

```python
if variables.input_value > 10:
    return "path_a"
else:
    return "path_b"
```

2. Based on what you return (like `"path_a"` or `"path_b"`), the flow will **follow the edge** with a matching label.

3. You can connect **two or more outputs** from the conditional edge to other nodes — each with a **label** matching the values returned from the code.

How to set up the connections

- After writing your conditional logic, **drag an edge from the output port** to the next node.
- You'll be prompted to **enter a condition label** (like `"path_a"` `"path_b"` `"default"` etc.).
- The flow will follow the edge that **matches the returned label** from your Python code.

Example

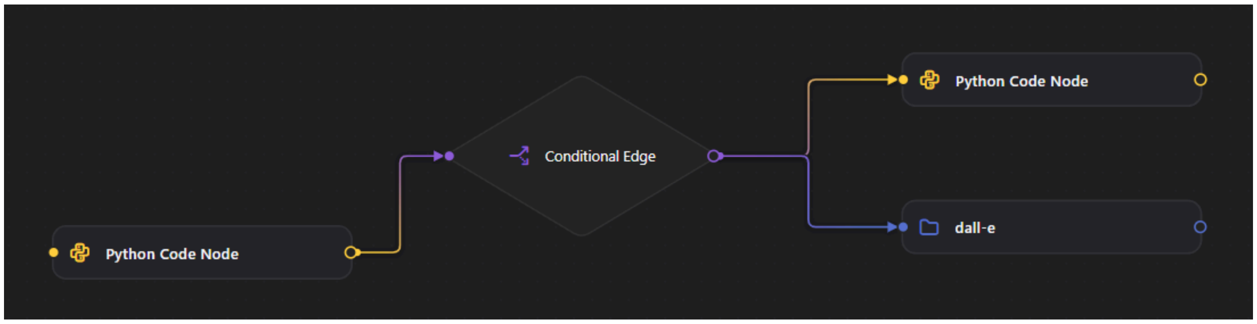
If your condition code is:

```Python
if variables.status == "approved":
    return "go"
else:
    return "stop"
```

You should connect the conditional edge to two nodes:

- One with the label `"go"`
- One with the label `"stop"`

The flow will automatically route to the correct one based on the logic.



---

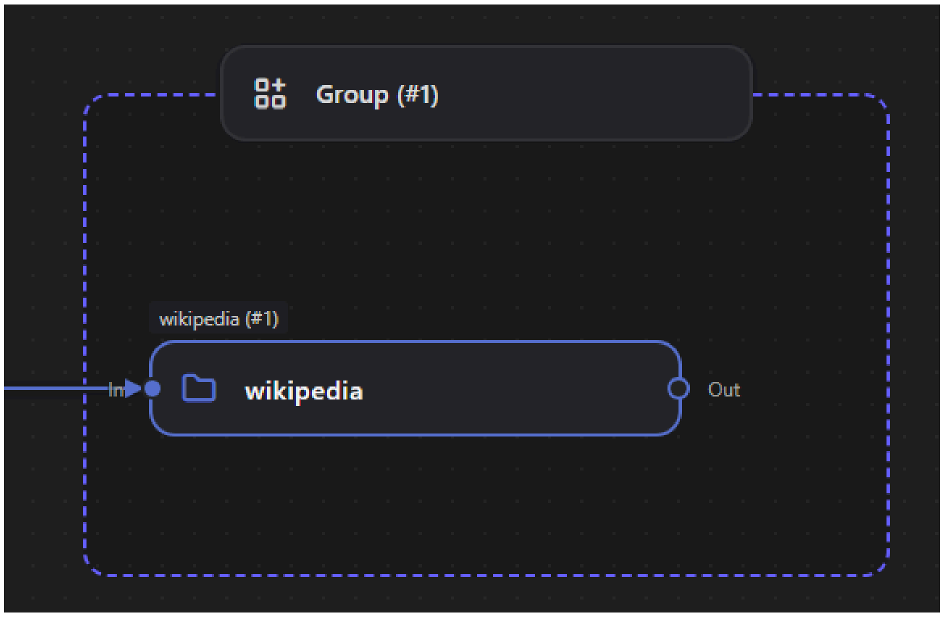
5. Group

A **Group** is a visual container inside your flow that helps organize nodes together. It's useful for keeping related logic in one place, especially in large flows.

How to use Groups

Add nodes to a Group:

- Just **drag and drop** any node into the group box.
- Once inside, the node becomes part of that group.



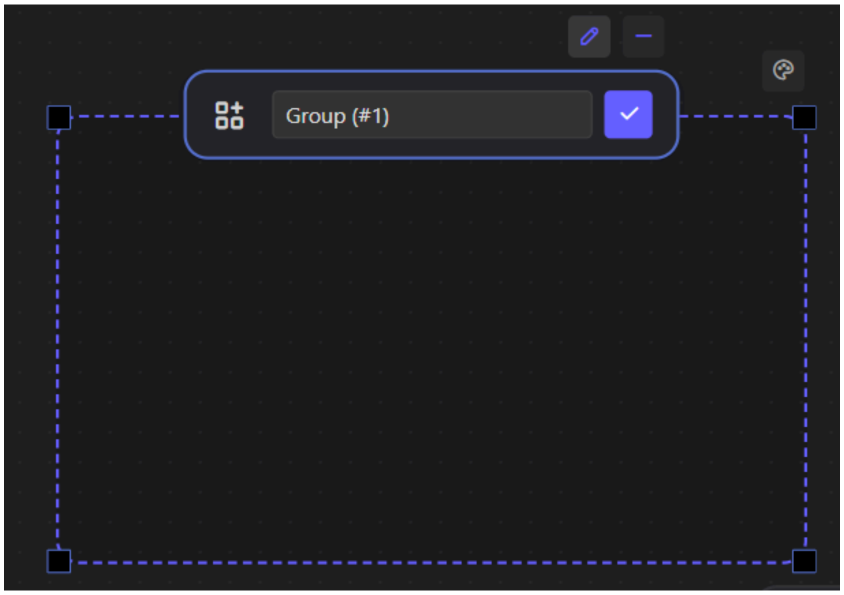
Remove a node from a Group:

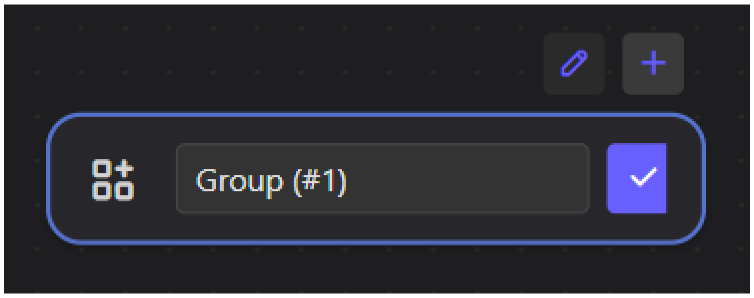
- **Hover over the node**, and you'll see a **special "remove from group" button** appear (usually near a corner).
- Click that button to take the node out of the group.

Group Options

You can customize the appearance and behavior of each group:

Feature	Description
Rename Group	Click the group name to edit it
Change Color	Use the color picker to change the background of the group
Collapse/Expand	Click the **+ / –** button near the group name to hide or show its contents





Why use Groups?

- Keep your flow clean and easy to understand
- Visually separate different logic sections
- Make large flows easier to debug and manage

What are variables in flows?

Variables are a way to **pass data between nodes** in a flow.

They act like named containers that store values (such as text, numbers, or outputs from a node), and they make it possible for different parts of your flow to share and reuse data.

Where and how are variables used?

Variables are used in multiple parts of the flow system:

1. Start Node

- This is where the flow begins.
- You can set **initial values for variables** here.
- Example:
  In the Start Node configuration, you might define:

```json
{
  "first_name": "Anna",
  "city": "Paris"
}
```

These variables can now be used by all other nodes.

More about nodes here: Nodes

---

2. Input Map

Every node (Python node, project node, conditional edge) has an **Input Map** field.

This is where you tell the node **which variables to use** as its inputs.

Example:

If your node expects a variable named `first_name`, your Input Map should look like this:

```json
{
  "name": "first_name"
}
```

Inside your Python code or project, you can now access the variable as:

```python
variables.name
```

More about nodes here: Nodes

---

3. Output Variable Path

If your node **returns a result**, you can store that result in a new variable using this field.

Example:

- You write Python code that returns a value like:

```python
return {"result": greeting}
```

- In **Output Variable Path**, you type:

```
variables.my_result
```

Now, a new variable called `my_result` will be available for the next nodes.

More about nodes here: Nodes

Passing variables between nodes

Once a variable is defined or updated in one node, it can be passed along and used in the **Input Map** of any other node in the flow. This creates a **chain of data** between nodes.

More about nodes here: Nodes

Using variables in projects via `{}`

When working with **Project Nodes**, you can pass variables into the agent's task prompt or project template using **curly braces** `{}`.

Example:

If you defined a variable `destination = "Rome"` in the Start Node, you can write this in your project task:

> "Find cheap flights to **{destination}**"

The system will replace `{destination}` with `"Rome"` automatically when the flow runs.

⚠️ This only works if:

- The variable was already defined in a previous node (like the Start Node)
- It is available under `variables.destination` in the flow

More on how to create and launch project here: Projects, Flows

---

Variables in python code (main)

Inside a Python Node, variables are always accessed like this:

```python
input_value = variables.my_variable
```

You should never hard-code values inside your `main()` function unless it's for testing or temporary usage.

Correct way:

```python
def main():
    name = variables.user_name
    return {"result": f"Hello, {name}!"}
```

Temporary (not recommended for final setup):

```python
def main():
    name = "John"  # Not using variables
    return {"result": f"Hello, {name}!"}
```

What happens if a variable is missing?

If you reference a variable that doesn't exist:

- The node might crash or return an error
- The flow could stop working at that point

Always make sure that:

- The variable is either defined in the **Start Node**
- Or it was created earlier in the flow using another node's **Output Variable Path**

Summary: Key Rules for Variables

Rule	Explanation
Variables must be passed using the `variables.` prefix	`variables.city`, `variables.user_age`, etc.
Input Map connects flow variables to node inputs	Tells the node what values to use
Output Variable Path stores a node's result	Makes it available to future nodes
In Project Node prompts, use `{}` to insert variables	Example: `"Hello, {first_name}!"`
In Python code, always access values through `variables.`	Not by hardcoding!
Variables are global inside the flow	Once created, they can be used by any future node

What is Human Input?

Human Input

pauses the flow

How to enable Human Input

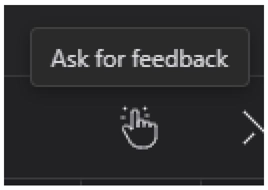
per task

Here's how:

Project Page

row

"Human Input"



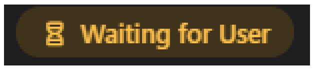
📌 **This is the only way to activate Human Input.**

[Projects](#) [Flows](#)

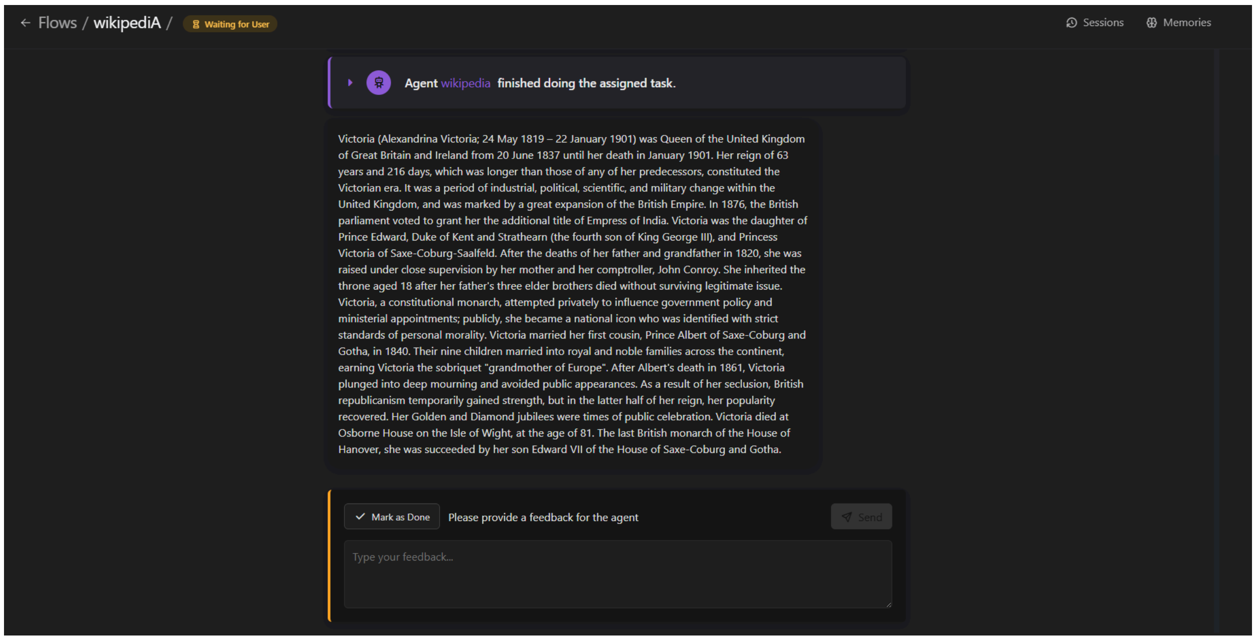
What happens during execution?

Human Input

pause



chat window

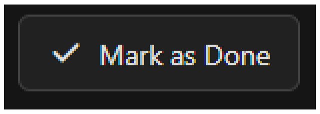


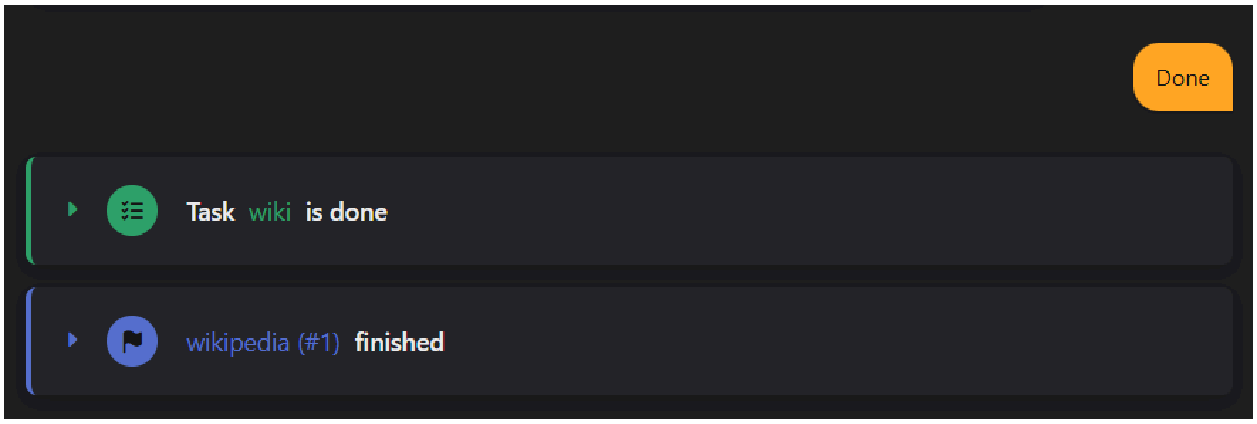
Agent wikipedia finished doing the assigned task.

Victoria (Alexandrina Victoria; 24 May 1819 – 22 January 1901) was Queen of the United Kingdom of Great Britain and Ireland from 20 June 1837 until her death in January 1901. Her reign of 63 years and 216 days, which was longer than those of any of her predecessors, constituted the Victorian era. It was a period of industrial, political, scientific, and military change within the United Kingdom, and was marked by a great expansion of the British Empire. In 1876, the British parliament voted to grant her the additional title of Empress of India. Victoria was the daughter of Prince Edward, Duke of Kent and Strathearn (the fourth son of King George III), and Princess Victoria of Saxe-Coburg-Saalfeld. After the deaths of her father and grandfather in 1820, she was raised under close supervision by her mother and her comptroller, John Conroy. She inherited the throne aged 18 after her father's three elder brothers died without surviving legitimate issue. Victoria, a constitutional monarch, attempted privately to influence government policy and ministerial appointments; publicly, she became a national icon who was identified with strict standards of personal morality. Victoria married her first cousin, Prince Albert of Saxe-Coburg and Gotha, in 1840. Their nine children married into royal and noble families across the continent, earning Victoria the sobriquet "grandmother of Europe". After Albert's death in 1861, Victoria plunged into deep mourning and avoided public appearances. As a result of her seclusion, British republicanism temporarily gained strength, but in the latter half of her reign, her popularity recovered. Her Golden and Diamond jubilees were times of public celebration. Victoria died at Osborne House on the Isle of Wight, at the age of 81. The last British monarch of the House of Hanover, she was succeeded by her son Edward VII of the House of Saxe-Coburg and Gotha.

Please provide a feedback for the agent

type replies as many times as needed
“Mark as Done”





Task wiki is done

wikipedia (#1) finished

[illegible]

[illegible]

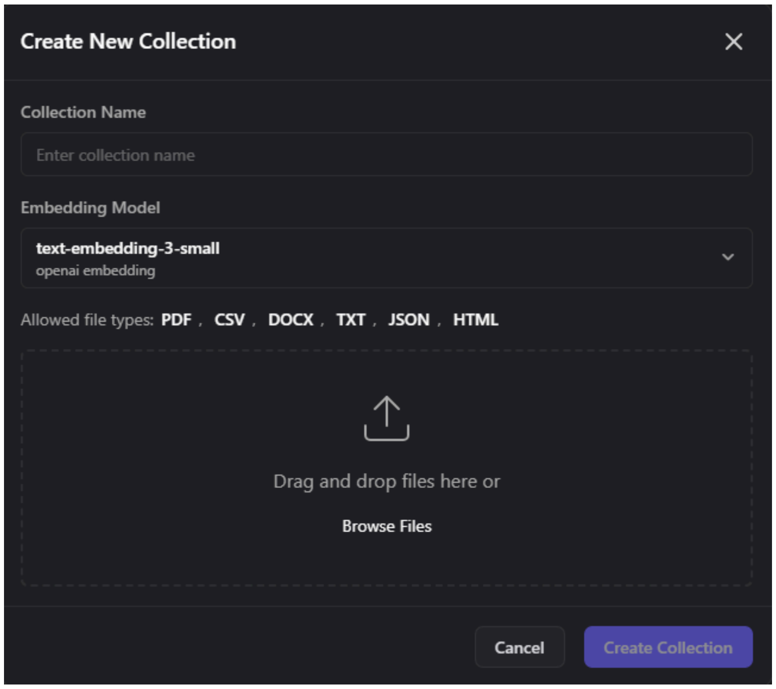
[illegible]

Agents can be enhanced with external documents and data through **Knowledge Sources**, allowing them to reference structured or unstructured information **before generating responses**. This [illegible]

[illegible]

[illegible]

[illegible]



Create New Collection

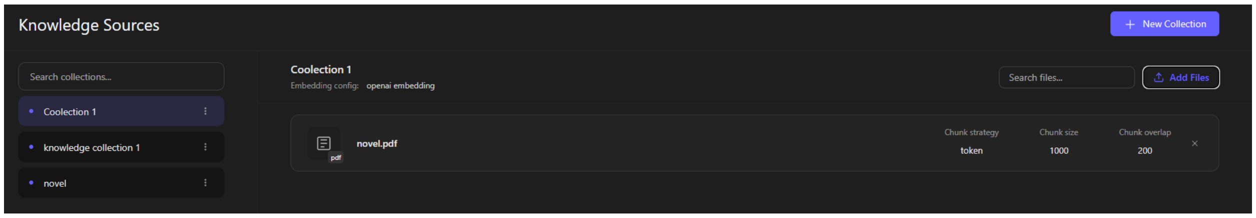
Collection Name

Enter collection name

Embedding Model

text-embedding-3-small
openai embedding

Allowed file types: PDF, CSV, DOCX, TXT, JSON, HTML



Knowledge Sources

Coolection 1
Embedding config: openai embedding

	Chunk strategy	Chunk size	Chunk overlap
novel.pdf	token	1000	200

File Settings: Chunking strategy

Once a file is uploaded, you'll configure how it's broken down ("chunked") for vector search.

Parameters:

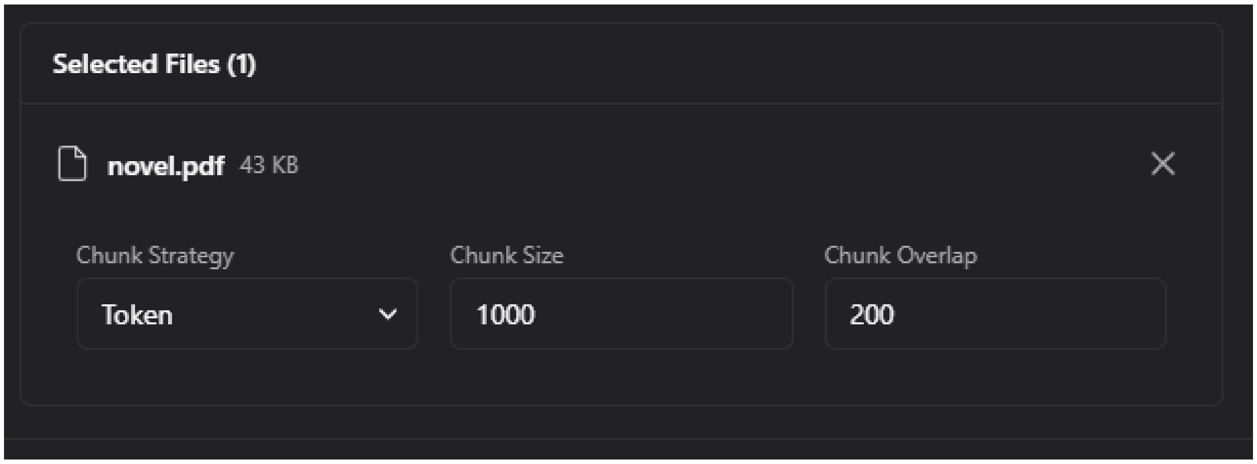
1. **Chunk Strategy** – Choose how to split the content:
    - **Token** – Splits based on number of tokens (language model units).
    - **Character** – Splits based on character count.
    - **Markdown** – Parses and splits using markdown structure (headers, lists).
    - **JSON** – Parses JSON objects.
    - **HTML** – Parses structured HTML content.

2. **Chunk Size** – The size of each chunk (in tokens, characters, etc., depending on the strategy). Affects how much content the model can reference at once.

Tip: Larger sizes = more context, but slower search.

3. **Chunk Overlap** – Number of tokens/chars that repeat between chunks. Prevents splitting mid-sentence or mid-paragraph.

   ⚠️ Must be smaller than chunk size.



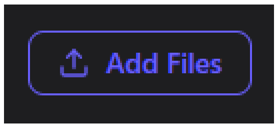
Selected Files (1)

novel.pdf 43 KB

Chunk Strategy: Token
Chunk Size: 1000
Chunk Overlap: 200

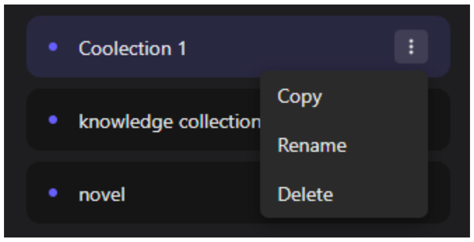
Managing files in a collection

- To **add more files** later:
  Click **Add Files** in the top-right corner of a collection.



- To **remove a file**:
  Click the **X** icon next to the file name.

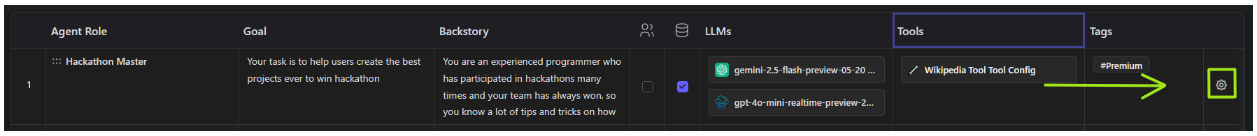
- To **rename or delete a collection**:
  Click the **three dots** beside the collection title.



Assigning a knowledge source to an agent

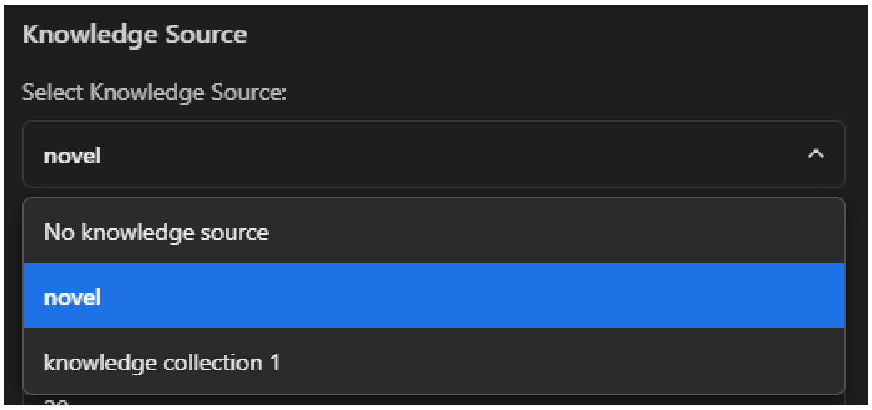
Once your collection is ready:

1. Open the **Agents** table.
2. Click the **gear icon (⚙️)** next to the agent you want to configure.



	Agent Role	Goal	Backstory			LLMs	Tools	Tags	
1	::: Hackathon Master	Your task is to help users create the best projects ever to win hackathon	You are an experienced programmer who has participated in hackathons many times and your team has always won, so you know a lot of tips and tricks on how	☐	☑	gemini-2.5-flash-preview-05-20 ... gpt-4o-mini-realtime-preview-2...	Wikipedia Tool Tool Config	#Premium	⚙

3. Scroll to the **Knowledge Source** section.
4. Under **Select Knowledge Source**, pick your desired collection from the dropdown.



Knowledge Source

Select Knowledge Source:

novel

No knowledge source

novel

knowledge collection 1

5. **Save** changes.

Now the agent will reference that collection during execution.

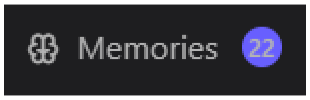
Read how to create agents here: [Agents]()

Why Use Knowledge Sources?

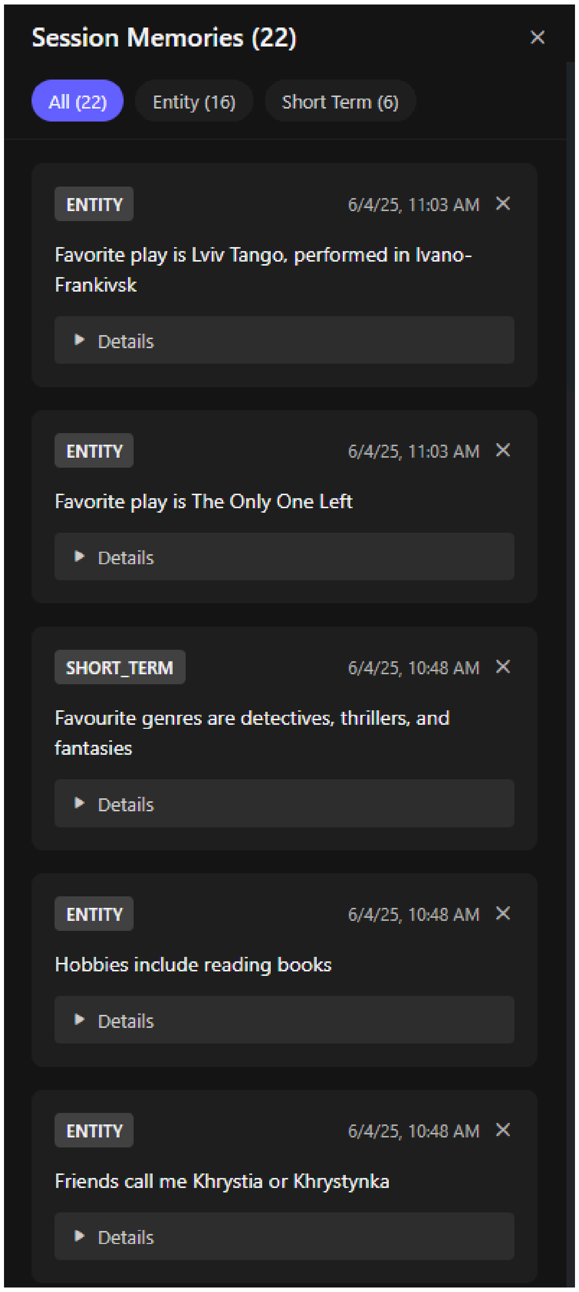
- Give agents access to internal data, guides, research, or instructions.
- Improve task accuracy and domain-specific reasoning.
- Avoid having to "teach" the same information in every prompt.

When your project is running in a flow (on the **Run Session Page**):

1. Look at the **top-right corner** of the screen
2. Click the **"Memories" button**



This opens a **sidebar panel** showing the memory contents for that session.



Session Memories (22)

All (22) Entity (16) Short Term (6)

ENTITY 6/4/25, 11:03 AM

Favorite play is Lviv Tango, performed in Ivano-Frankivsk

Details

ENTITY 6/4/25, 11:03 AM

Favorite play is The Only One Left

Details

SHORT_TERM 6/4/25, 10:48 AM

Favourite genres are detectives, thrillers, and fantasies

Details

ENTITY 6/4/25, 10:48 AM

Hobbies include reading books

Details

ENTITY 6/4/25, 10:48 AM

Friends call me Khrystia or Khrystynka

Details

Types of memory

There are **four types of memory**, each serving a different purpose:

1. Short-Term Memory

- Automatically stores **temporary thoughts or facts** gathered by the agent during reasoning.
- Useful for immediate next steps.
- Example: intermediate steps in solving a problem.

2. Long-Term Memory

- Automatically stores **important knowledge** the agent decides is worth keeping during the session.
- Typically higher-level conclusions, goals, or facts.

3. Entity Memory

- Tracks **structured information** like:
  - Names
  - Dates
  - Locations
  - Other extracted data from text

- Example: if the user says "My name is Olivia", the system may store:

```
None
name: Olivia
```

4. User Memory

- Only created when a user **types something manually** using a Human Input node.
- Stores whatever the user enters, so it can be referenced later.

More about Human input here: Human Input

Interacting with Memory

From the memory sidebar, users can:

- **View each memory entry**
- **Delete** any specific memory item
- **Filter** by memory type (or view all combined)

Summary

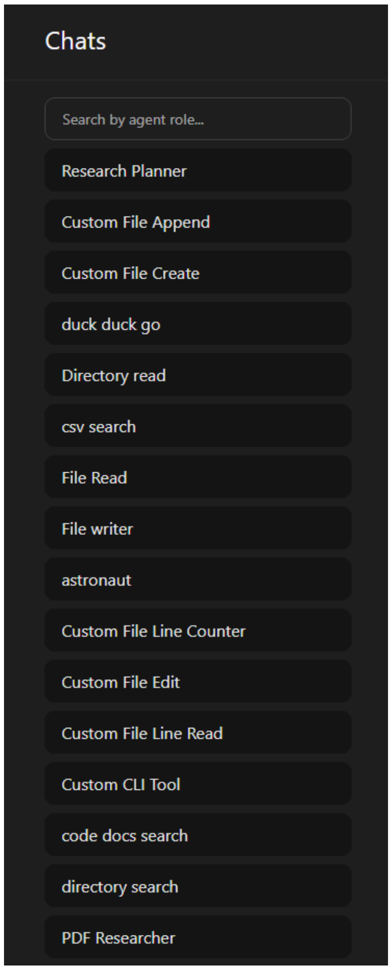
Feature	Description
Project Memory	Active, session-based, enables agents to remember info
Agent Memory	Not available yet
Activation	Enable in project settings
Access	Available via “Memory” button during a flow run
View/Edit	You can read or delete memory entries from the sidebar
Types	Short-Term, Long-Term, Entity, User Memory

The **Realtime Chat** page allows you to talk to your agents **via voice or text** in real time. This feature is useful for testing agents interactively or having a natural conversation with them.

Agents Displayed

On this page, you will see **all agents you've created** in the system – the same ones listed in your **Staff Table**. (Read how to create and configure agents here: [Staff](#))

⚠️ **Only agents with a configured Realtime LLM model can respond in this chat.**



Chats

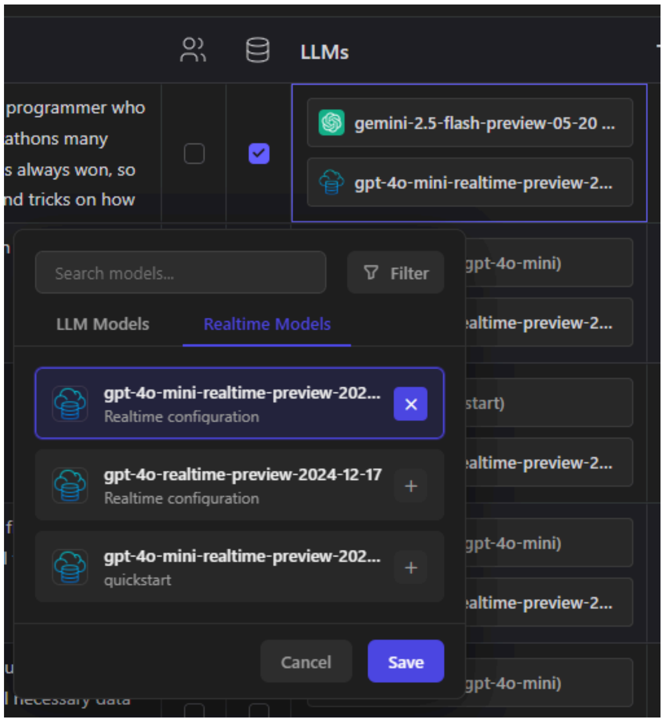
Search by agent role...

- Research Planner
- Custom File Append
- Custom File Create
- duck duck go
- Directory read
- csv search
- File Read
- File writer
- astronaut
- Custom File Line Counter
- Custom File Edit
- Custom File Line Read
- Custom CLI Tool
- code docs search
- directory search
- PDF Researcher

Requirements to Enable Voice Chat

Before an agent can respond in Realtime:

1. **You must assign a Realtime LLM model**
   - Go to the **Staff Table**
   - Double-click the `LLMs` cell of your agent
   - Open the **Realtime Models** tab in the popup
   - Click the ➕ button to assign a Realtime model (only one can be active)

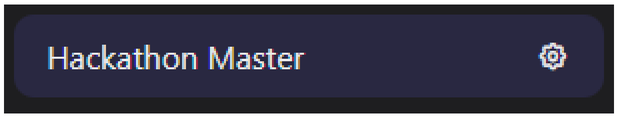


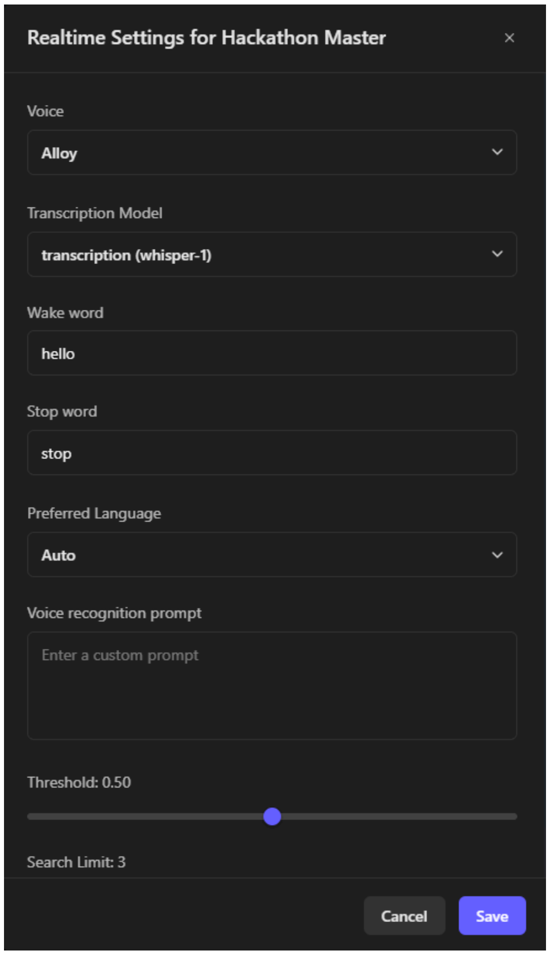
LLMs
gemini-2.5-flash-preview-05-20 ...
gpt-4o-mini-realtime-preview-2...
Search models...
Filter
LLM Models
Realtime Models
gpt-4o-mini-realtime-preview-202...
Realtime configuration
gpt-4o-realtime-preview-2024-12-17
Realtime configuration
gpt-4o-mini-realtime-preview-202...
quickstart
Cancel
Save

A detailed instructions on how to set up a realtime LLM is here: LLMs

2. **Agent must be configured in the Realtime Chat page**
   Next to the agent's name, click the **gear icon** to open their **Realtime Settings**.

Realtime Agent Settings





Realtime Settings for Hackathon Master

Voice: Alloy

Transcription Model: transcription (whisper-1)

Wake word: hello

Stop word: stop

Preferred Language: Auto

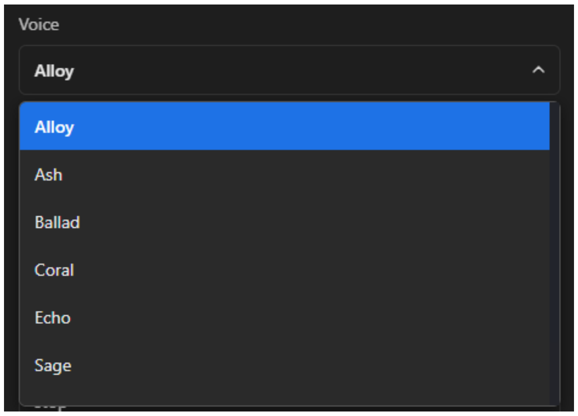
Voice recognition prompt: Enter a custom prompt

Threshold: 0.50

Search Limit: 3

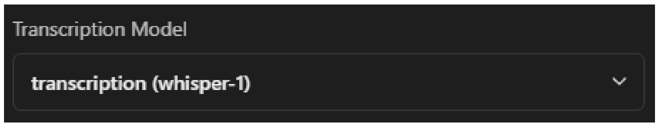
In the Realtime Settings modal, you'll see:

- **Voice**: Choose a voice profile (cosmetic only – it doesn't affect functionality)



Voice: Alloy

- **Transcription Model** ( ):
  Used to convert your speech to text. Without it, the agent won't hear or respond.

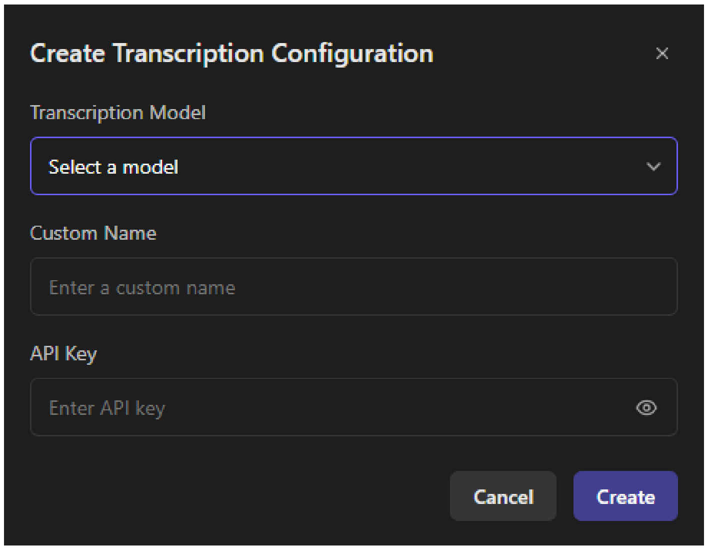

Transcription Model

transcription (whisper-1)

If no Transcription Model exists:

- Click the dropdown → **Create New Configuration**

- Fill out the modal form:
    - **Transcription Model**: Select one from the list (dropdown)
    - **Custom Name**: Give it a name
    - **API Key**: Add your service key if required


Create Transcription Configuration

Transcription Model

Select a model

Custom Name

Enter a custom name

API Key

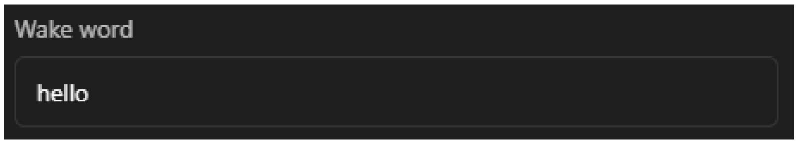
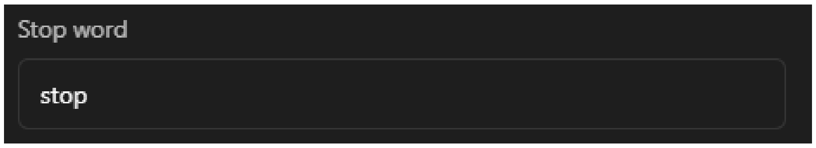
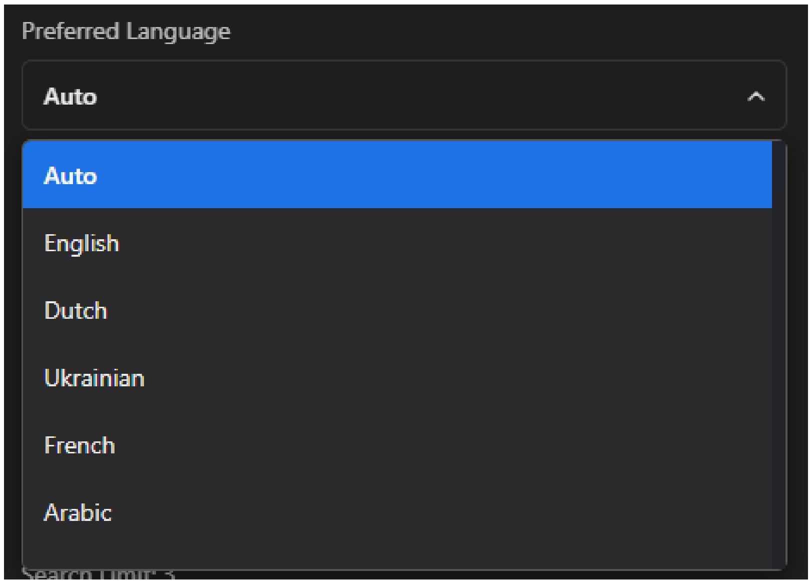
Enter API key

- Once the required fields are filled, the **Create** button becomes active.
- Click it to save and select your new configuration.

Additional Realtime Settings (Optional)

These options are shown in the agent's Realtime Settings window:

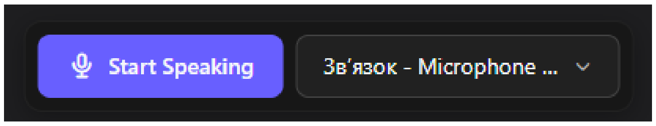
Field	Description

Wake Word	A word that activates the agent (e.g., "Hey assistant", “Hello”, Agent’s name)  Wake word: hello
Stop Word	A word that stops the agent from listening  Stop word: stop
Preferred Language	Language used for voice recognition  Preferred Language: Auto
Voice Recognition Prompt	A hint to improve transcription accuracy (e.g., “You are listening to technical commands”). Also good practice to mention agent wake word here
Threshold (0–1)	Threshold for searching by knowledge. Lower = more accurate knowledge will be extracted. Default: 0.6
Search Limit (0–1000)	Limits how many knowledge chunks will be extracted. Default: 3

How to Start Talking to Your Agent

Once the LLM and transcription settings are configured:

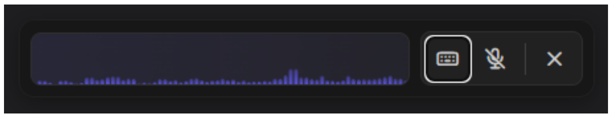
1. **Allow microphone access** in your browser when prompted
2. Select the agent you want to talk to
3. Click **Start Speaking**

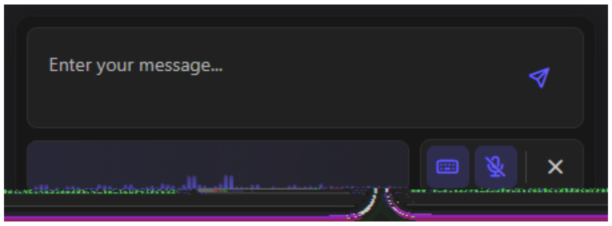


Chat Interface Controls

During the chat session, you'll see:

Icon	Function
Microphone (when white)	Tap to speak – the system listens and responds
Microphone (when purple)	Click to mute yourself; the agent will **not** listen
Keyboard	Click to type a message instead of speaking
X (Close Chat)	Ends the conversation – **all messages will be lost**





⚠️ **Note**: Realtime chats are **not saved**. Once the session is closed, all messages are gone.