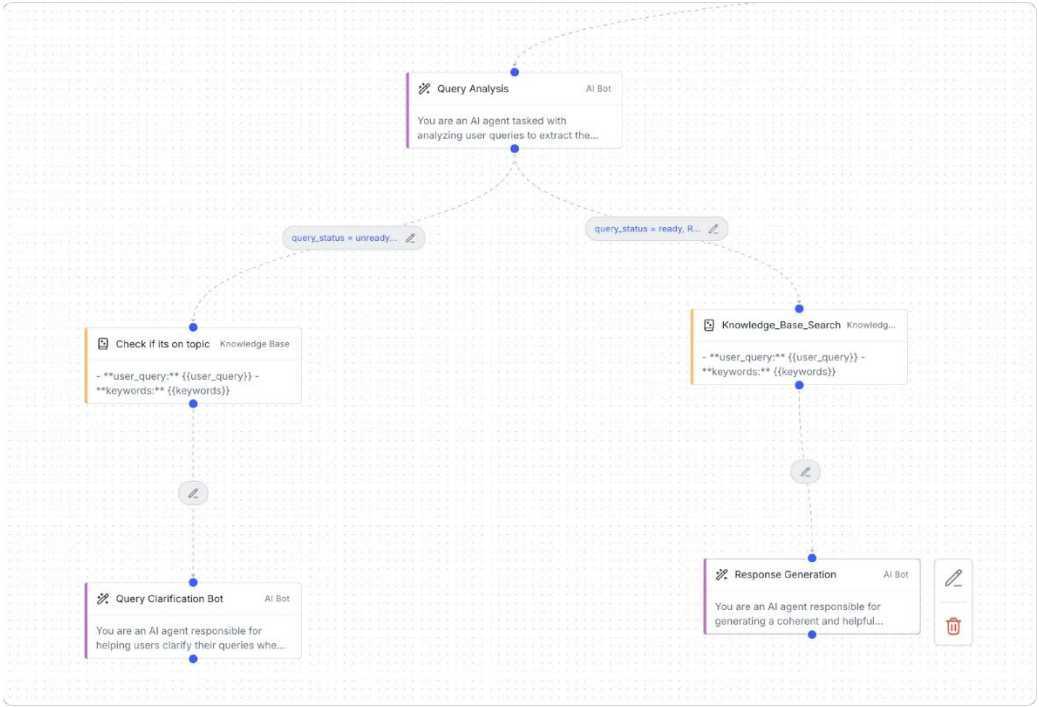


Note: AI Tool. Use at own risk.

For authorized Clerk Chat Workspace Owners or Administrators using the Ultimate license package, Clerk Chat's Agents provides access to use the power of AI with tools that allow you to create Flows that can respond to contacts automatically. A Flow is a collection or series of individual AI bots you create that can collaborate and perform predefined tasks in Clerk Chat. For example, you could create a collection of AI bots that work together to answer FAQs from your knowledge base. Those bots would be designed to understand a question, search for the answer, generate the answer, and send that answer back to the user as a text message - like a human Agent.

What is an Agent?

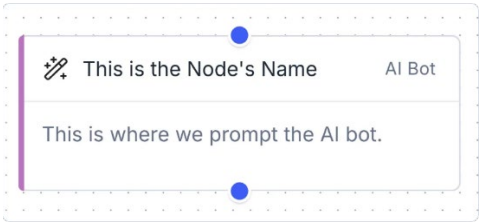
An agent is a series of individual AI bots (nodes) that work together in a flow to perform tasks.



Key components of an Agent

An agent is made up of Nodes.

A node is one AI bot. When a node is triggered, it takes whatever is in the prompt section, sends it to ChatGPT and the response is returned to your Clerk Chat.



There are a few sections to each Node that are important:

Edit node

Information

Type

AI Bot

Name

This is the Node's Name

Triggered By

User message

Response Type

Bot message

Prompt

Prompt to be used for the AI Bot.

This is where we prompt the AI bot.

Type

- AI Bot: Has a prompt section. This sends whatever is in the prompt section to ChatGPT.
- Knowledge Base: These nodes can search a team’s knowledge base (in the Knowledge Base section) and output the response as a message so that it can be used by other nodes in the pipeline.
- AI Bot Tool Runners: These nodes provide tools to the AI Bots. Each tool runner provides one or more tools (e.g searching a calendar or creating calendar event).

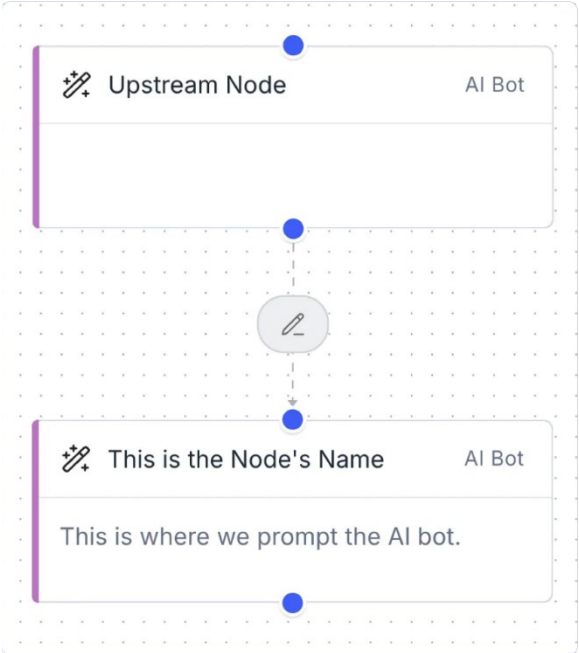
Name

The title of the node.

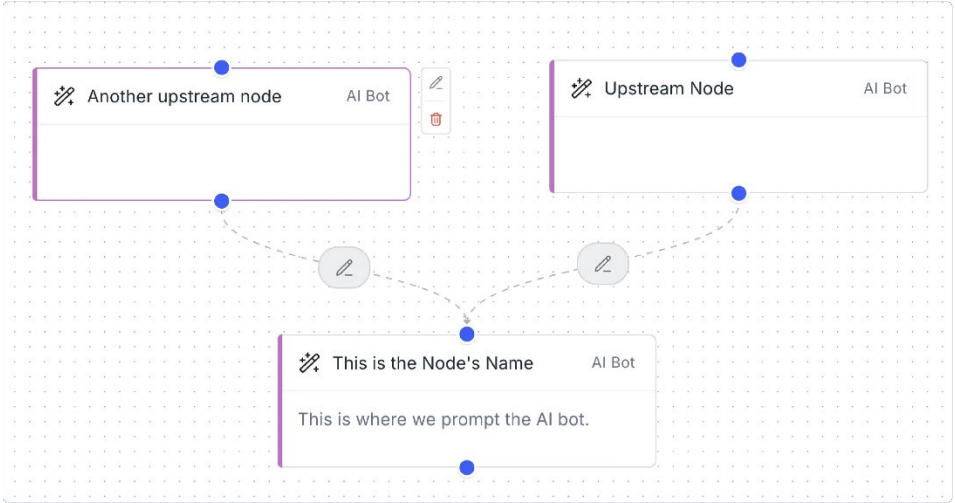
Triggered By

What triggers the node (causes it to activate/run)?

- User message: Triggered by a user sending a text message to the line the agent is enabled on.
- Scheduled Run: Triggered by a scheduled run (using the follow-up tool).
- Unread message from any: Node will be started when **any** of the upstream nodes in the pipeline sends it a message. I.e. just need one upstream node to send a message to be triggered.



- Unread message from all: Node will be started when **all** of the upstream nodes attached have sent it a message. i.e. in the example below, the bottom node won’t trigger until it has received a message from both upstream nodes.



Response Type

- JSON Messages: The **AI Bots** can be instructed to return JSON.
 - My name is William Bowen, I work at Clerk Chat and I have brown hair.
 - Name: William Bowen
 - Company: Clerk Chat
 - Hair: Brown
- User Messages: Regular text (natural language) that will be sent to the user over SMS.

Tip - If you want to send a message to another AI bot, use JSON message. It forces some structure in the agent.

If you want to send a message back to the user over SMS, then use user message.

Try not to use *bot message* as it offers less structure to the agent (meaning things are harder to understand and debug down the line).

Prompt

This is where we give the AI bot its prompt. It is very good to follow a structure for ALL prompts you write. Here's an example template:

```
# Role:

---

# Goal:

---

# Context:

---

# Conversation History
{{#conversationHistory}}
{{#isUser}}Client{{/isUser}}{{^isUser}}Clerk_Chat{{/isUser}}: {{content}}
{{/conversationHistory}}

---

# Company Overview

---

# Instructions:

---

# Example input and output:
```

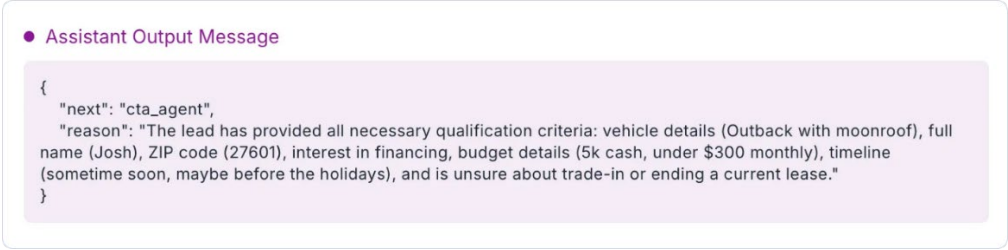
The 'Conversation History' section is a blob of JSON that takes the complete SMS conversation history and injects it into the prompt. If you copy the above prompt with the conversation history, the AI will have access to the complete conversation every time it executes which is pretty useful.

Extract Variables

Extract variables allow data to be shared between nodes. Extract variables are created when a node has a JSON Response type.



Here’s an example of a node with two extract variables. In this case, the node will execute with the prompt, ChatGPT will send a response back and the response will be mapped to these two variables.



This is what the response looked like from this node in a live example. You can see it outputs a JSON message that includes only the two extract variables specified.

Extract variables can be used in any downstream nodes in the pipeline.

An example of this is in an FAQ bot. Where the complete agent understands a user question, searches a knowledge base and returns the answer.

In this case, one bot would extract the user question in a JSON response.

This would create an extract variable `user_question`.

This variable can then be passed to the bot that does a search for the answer. It can search for `user_question` and return the answer.

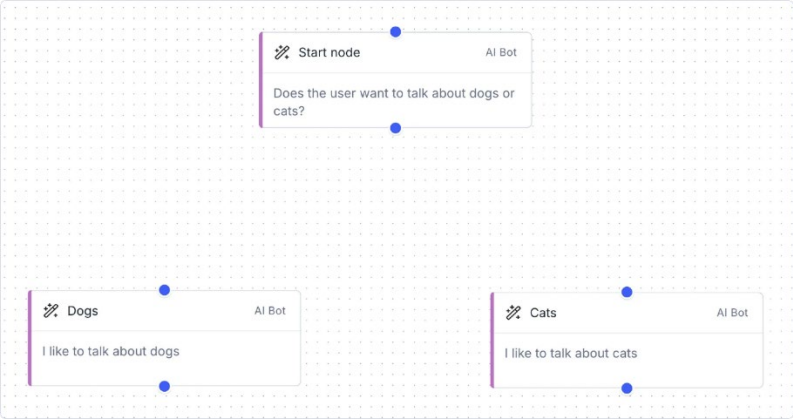
Edges

Edges can do basic filtering. This is where the real magic of the Agents section comes in.

Edges connect the nodes together. When a node outputs a message, the pipeline runner uses the node’s edges to determine which nodes should receive the message. Each edge has a ‘source’ and a ‘destination’.

Example: I have an agent that can either talk to users about dogs or cats. How does the agent decide if the contact wants to discuss dogs or cats?

This is where Edges come in.

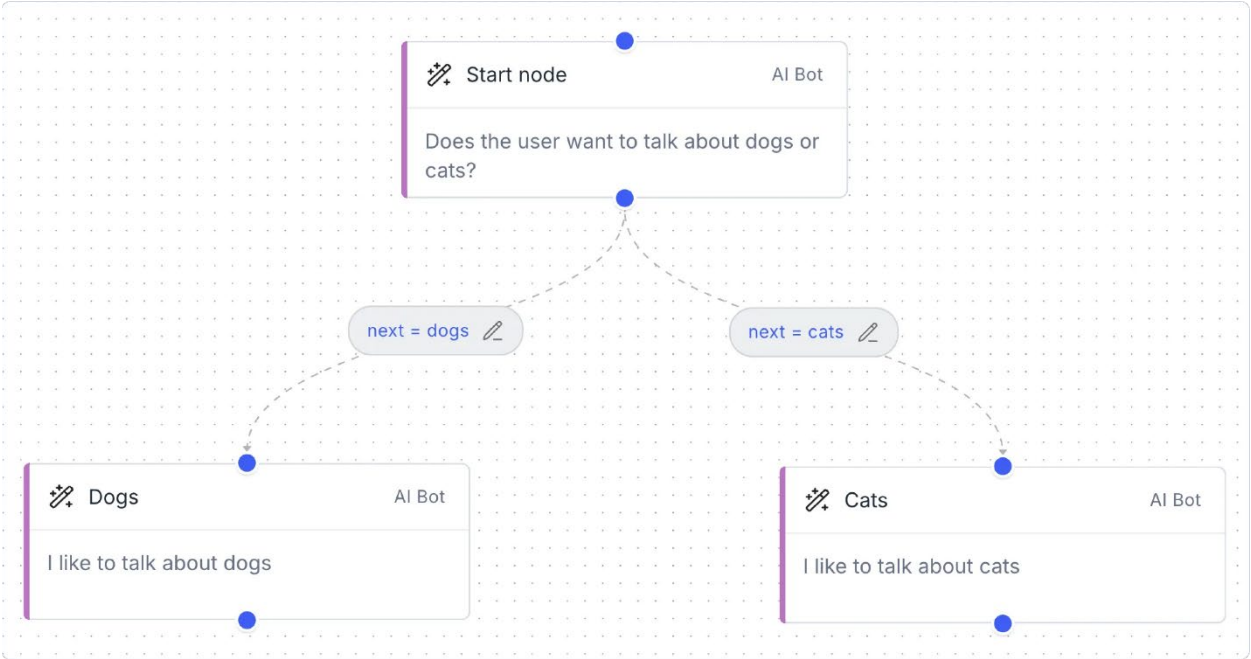


We can use edges to choose the next node that is activated using extract variables to filter on edge nodes.

In this case, the extract variable is `next`

In this case, if the start node outputs: `next = dogs`, then the next node to be activated is the Dogs node.

If the start node outputs: `next = cats`, then the next node activated is the Cats node.



This makes the agents section powerful as we can branch the AI bots and create more complex logic to solve a variety of tasks.

Building an Agent

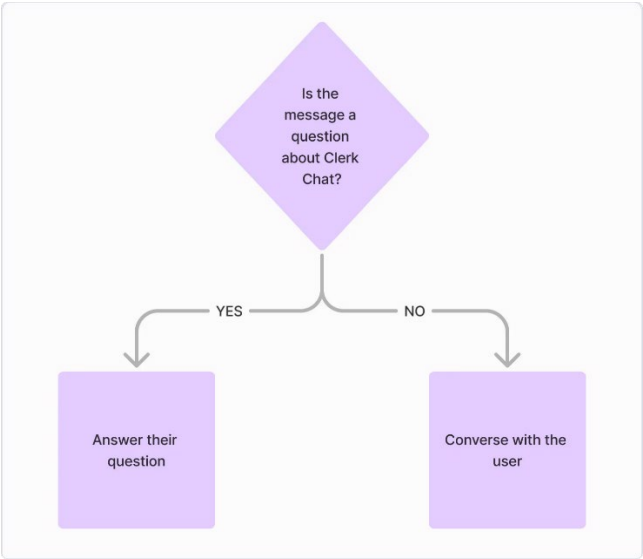
How do we build an agent? Let’s go step by step through the process of building a useful FAQ flow in Clerk Chat.

- Goal** - Create an Agent that converses with a user and answers their FAQs on Clerk Chat
- Logic** - Before building in Agents, we need to map out the logic.
This is where we figure out how we want the AI to think about this problem.

The ‘on-topic’ conversations will be asking a question about Clerk Chat. In this case, we want the AI to send back an answer.

If the message is not a question about Clerk Chat, then we just want to AI to converse back and forth with the user.

Here’s a diagram of that in *Figma* (a great 3rd party tool for mapping out AI logic).



This is a great starting point.

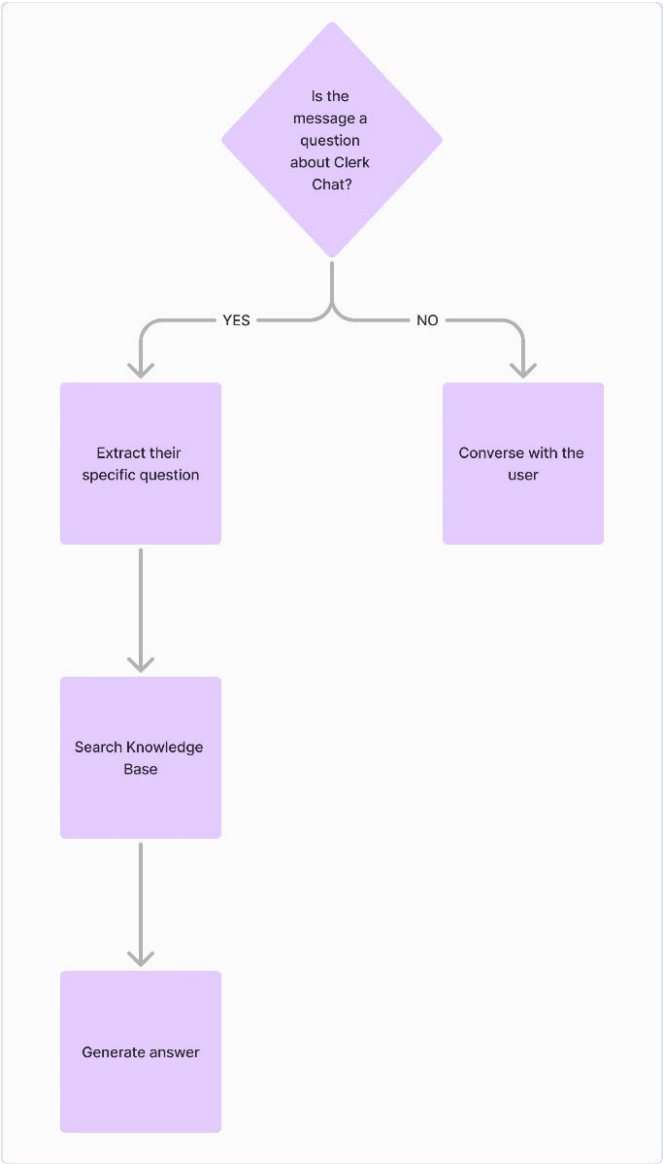
We can break it down a bit further by asking, “how would the AI answer their question?”

The AI would have to:

1. Extract the user’s question (it may be in a sentence and we just want to search for the answer to their specific question, not the whole sentence).
2. Search the knowledge base for the answer.
3. Generate an answer.
4. Send it back to the user.

What about the ‘Converse with the user’ side of the agent?

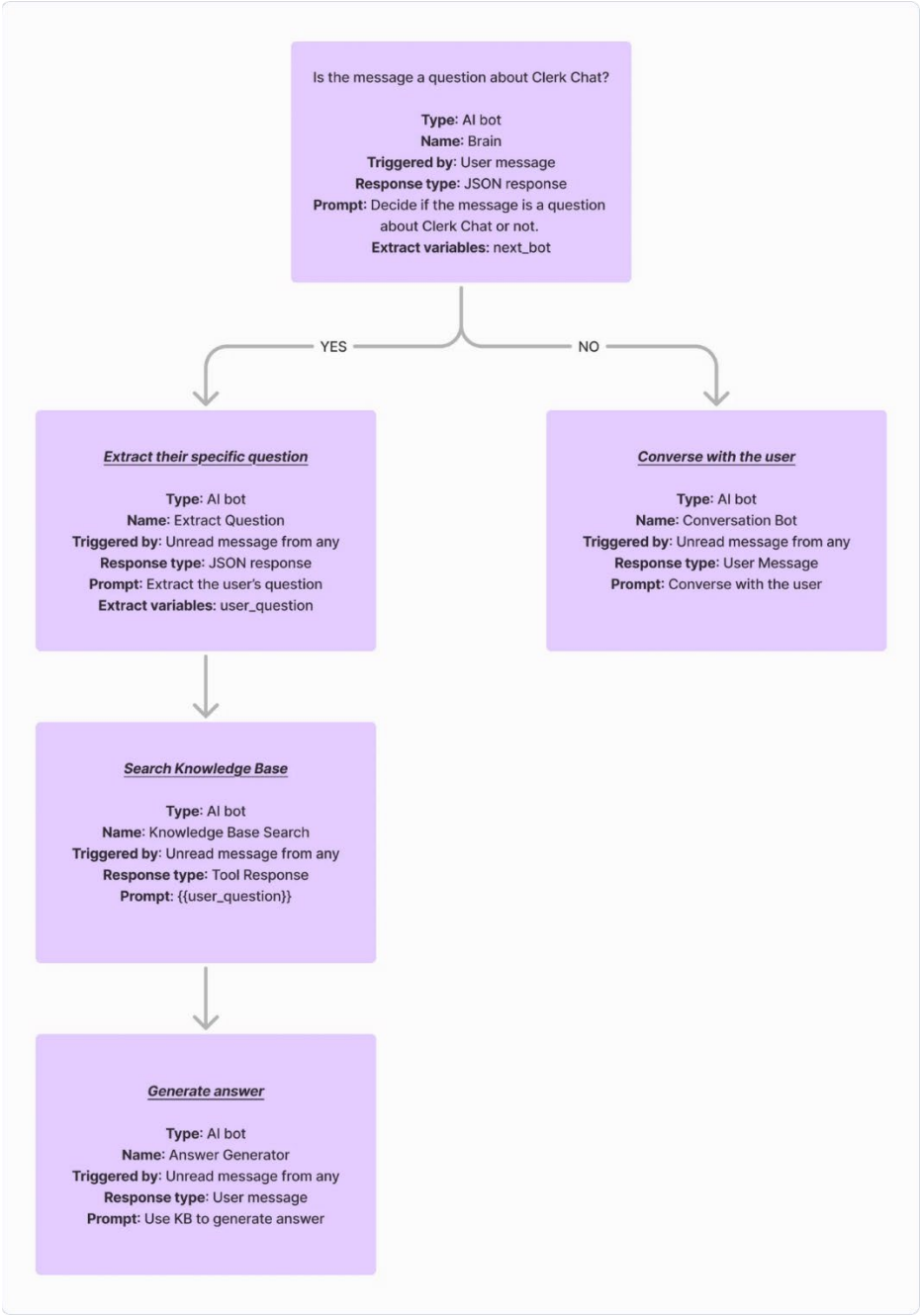
In this case, the AI just has to read the conversation, generate an appropriate conversational message, and send it back to the user via text. So our updated logic diagram might look like:



Finally, let’s add a bit more info to this diagram to map out some of the properties of each bot before we begin building in Agents.

- Type
- Name
- Triggered by
- Response type
- Prompt
- Extract variables (if any).

This is the logic diagram we might end up with:



Explanation:

- We only want the top (first) node to be triggered by a user message. We want all other nodes to be triggered only by nodes above them.
- We only want the last nodes in each branch to send an SMS message back to the user.
- Nodes that don't send a user message are set to JSON response as this allows for the most manageable structure.
- The Knowledge Base Search node is a tool call. A tool is a connection to something other than the AI. In this case, a database of documents. When calling a tool, there is always a tool response.

Now we are ready to start building in the Agents section.

Building Basics

Since we have a logic flow prepared, we can copy the logic and properties we have mapped out already. And then we can add a structured prompt using the prompt template. Here's an example:

```
# Role:

---

# Goal:

---

# Context:

---

# Conversation History
{{#conversationHistory}}
{{#isUser}}Client{{{/isUser}}{{^isUser}}Clerk_Chat{{{/isUser}}: {{content}}
{{/conversationHistory}}

---

# Company Overview

---

# Instructions:































---

# Example input and output:
```

Once you build an agent end to end, it's time to test it. Testing is the **final** part of building an Agent.

Clerk Chat Agent Creation Help

Here are some useful ideas from Clerk Chat for building various types of Agents.
To learn more, click on an item that interests you below to open the topic in your browser:

 What are Agents?	>
 Building an Agent	>
 FAQ Agent	>
 Follow Up Agent	>
 Qualification Clerk	>
 Qualification + Follow Up Clerk	>
 Write contact data	>
 Human Handoff Feature	>
 Contact Property Update Agent	>
 Sentiment Analysis Agent	>
 Urgent Agent Template	>
 Logistics Agent	>
 Property Management Repairs Agent	>
 Sales - useful information agent	>
 Support - CSAT Agent	>
 Customer success - feedback agent	>
 Sports teams - hype agent	>
 Gyms - sign-up agent	>
 Gym - re-engagement agent	>
 Airbnb - guide agent	>
 Church - new sign up agent	>
 HR - PTO agent	>
 Property management - qualifying agent	>
 Restaurant - takeaway agent	>
 Extract full name and email	>
 Answer questions from a knowledge base	>
 Auto response	>
 HTTP Request Workflow	>
 Assign to team member workflow	>
Voice AI - Knowledge base agent	>
 Build ANY workflow	>