

Google Cloud Skills Boost for Partners

[>Main menu](#)

Integrate Vertex AI Search and Conversation into Voice and Chat Apps

Course - 5 hours 20% complete

[Course overview](#)

Integrate Vertex AI Search and Conversation into Voice and Chat Apps

Build Vertex AI

Search Apps using AI Applications

Enable informed decision making with a conversational agent that uses generators and data stores

Course > Integrate Vertex AI Search and Conversation into Voice and Chat Apps >

Quick tip: Review the prerequisites before you run the lab

[Start Lab](#)

01:30:00

Lab instructions and tasks

-/100

Enable informed decision making with a conversational agent that uses generators and data stores

Lab 1 hour 30 minutes No cost Intermediate

Rate Lab

This lab may incorporate AI tools to support your learning.

Overview

Setup and requirements

Task 1. Enable APIs
Task 2. Create a new conversational agents and a data store for your app

Task 3. Configure the agent to answer blood eligibility FAQs
Task 4. Test the agent

Task 5. Set up the agent for the eligibility quiz
Task 6. Retest your agent

Task 7. Conclusions and next steps

[Next >](#)[Previous](#)

Overview

This lab serves as a comprehensive guide to building Conversational Agents using Vertex AI Agent Builder. It walks you through the process of configuring agents in the new Conversational Agents console, leveraging Playbook-based, flow-based approaches.

Key concepts you will learn in this lab:

- Vertex AI Agent Builder will be the starting point for building Conversational Agents.
- The new Conversational Agents console, which builds on the Dialogflow CX console, will be used for most agent configurations.
- Conversational agents can be Playbook-based, flow-based, or combination agents.
- Data stores are repositories of data, created in Vertex AI Agent builder, that are

the agent can parse user input and generate agent output using the model.

What you'll build

In this lab, you use Vertex AI Agent Builder and Conversational Agents to build, deploy and configure a conversational agent to assist people who want to donate blood and ensure they meet the required eligibility requirements. The agent uses real public data and Google's generative large language models (LLMs) during Conversational Agents fulfillment.

Conversational Agents

Conversational Agents is a new natural language understanding platform built on generative models that can control conversations and on flows that can be used for more explicit conversation control. Conversational Agents makes it easy to design and integrate a conversational user interface into your mobile app, web application, device,

[Data stores](#) are used by data store handlers and playbook data store tools to find answers for end-user's questions from your data. Data stores are a collection of websites and documents, each of which references your data.

[Data Store Settings](#) are the configurations that define how a conversational agent interacts with the data stores.

The [Vertex AI Agent Builder](#) feature allows you to create conversational agents powered by data stores.

With this feature, you provide a website URL, structured data, or unstructured data (data stores), and Google parses your content to build a conversational agent that uses the data from these stores and large language models. The agent can then interact with customers and end users, allowing them to ask questions and get answers based on the provided content.

Generators

Google's latest generative large language models (LLMs) and custom prompts to generate agent responses at runtime. A generator can handle generic responses that involve general knowledge from a large textual dataset it was trained on or context from the conversation.

Objectives

In this lab, you learn how to perform the following tasks:

- Use Conversational Agent to create a app and add unstructured data to a data store.
- Use knowledge handlers to allow end-users to have conversations with a conversational agent about the content added to a data store.
- Configure a generator text prompt and make it contextual by using built-in generator prompt placeholders.
- Mark words as generator prompt placeholders and later associate them with session parameters in fulfillment to use their values during execution.

Context, dataset, and context from the current conversation.

- Generate a formal email using generators.
- Test your agent and simulate customer questions that trigger generated responses.

Setup and requirements

In this task, you will use Qwiklabs to perform the initialization steps for your lab.

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Make sure you signed into Qwiklabs using an **incognito window**.

finish in that time block.

There is no pause feature. You can restart if needed, but you have to start at the beginning.

3. When ready, click **START LAB**.

4. Note your lab credentials. You will use them to sign in to the Google Cloud

Open Google Console

Cautions: When you are in the console, do not deviate.

ACCOUNT TO BE BLOCKED. [Edit more](#).

Username

google2876526_student@qwiklabs.n



Password

TG959yrKDX



GCP Project ID

qwiklabs-gcp-0855e773352d3560



New to labs? [View our introductory video!](#)

5. Click **Open Google Console**.

6. Click **Use another account** and copy/paste credentials for **this lab** into the

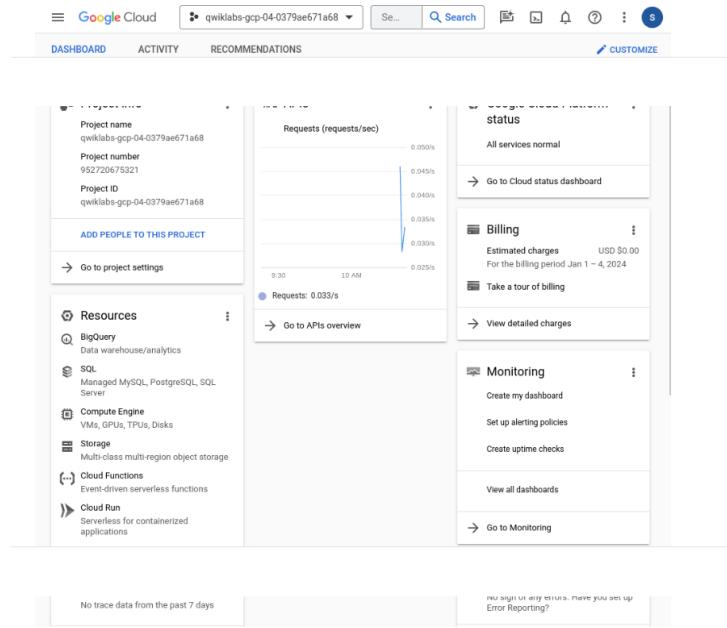
If you use other credentials, you'll get errors or **incur charges**.

7. Accept the terms and skip the recovery resource page.

Do not click **End Lab** unless you are finished with the lab or want to restart it. This clears your work and removes the project.

Google Cloud console

After you complete the initial sign-in steps, the project dashboard appears.



Ensure your project, <filled in at lab start> is selected. If not, then click **Select a project** at the top of the console, highlight your **Google Cloud project ID**, and click **OPEN** to select your project.

If you are not using Incognito mode, you will need to ensure that the correct project and user account are selected at all times to avoid accidental charges to your Google Cloud billing account.

Task 1: Enable APIs

Before you can start using Conversational Agent in [Vertex AI Agent Builder](#), you need to enable the Dialogflow as well as the Vertex AI Agent Builder APIs. These APIs should already be enabled for this lab, but you should verify this before moving forward.

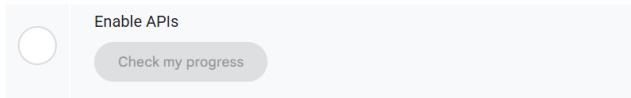
Enable the **Dialogflow API** by doing the following steps:

1. In your browser, navigate to the [Dialogflow API Service Details page](#).
2. If the API is not already enabled, click the **Enable** button to enable the Dialogflow API in your Google Cloud project.

To enable the **Vertex AI Agent Builder API**, follow these steps:

3. In the Google Cloud console, navigate to the [Vertex AI Agent Builder](#).

4. If asked, read and agree to the Terms of Service, then click **Continue and activate the API**.



Task 2. Create a new conversational agents and a data store for your app

Now, you'll create a new conversational agents for your agent and configure it with a data source. The purpose of the agent that you'll build is to assist customers who have questions about blood eligibility. You will use the Australian Red Cross Lifeblood as the

1. To create a new conversational agent, in [Vertex AI Agent Builder Apps Console](#) choose **Conversational agent** as the app type. Click **Create**.

2. Click **Create agent**.

3. Select **Build your own** on Get started with Conversational Agents popup.

4. On Create agent page, enter a **Display name** as **Blood Donation Agent** for agent.

5. Select location as **global (Global serving, data-at-rest in US)**.

6. Ensure the default **Conversation start** type is selected as **Playbook**, and then click on **Create** button.

7. Click **+ Data store** under Available tools in **Default Generative Playbook** page.

9. Click **Create new data store** on Add data stores page, it will be redirected to the Agent Builder page

10. Click **Select** on the **Cloud Storage** as the data source for your data store.

11. Select **Unstructured documents** as the type of data you are importing.

12. Specify the following Google **Cloud Storage** folder that contains sample data for this lab, and note that the `gs://` prefix is not required:

cloud-samples-data/dialogflow-cx/arc-lifeblood



13. Click **Continue**.

14. Specify a **Data store name** of **Australian Red Cross Lifeblood**

15. Click on **Create** to create the data store.

Note: This may take up to 1 minute to create the data store.

16. In the list of data stores, select the newly created data store named **Australian Red Cross Lifeblood Unstructured**.

17. Click on the **Activity** tab to see the progress of the data import.

Note: Generally, it can take up to 4 hours for your documents to be available and ready for use by your agent while your newly added domains are being indexed depending on the number of documents or size of your domain. For this lab, it will take 10-15 minutes.

19. Enter **Blood_donation_tool** in the **Tool name** field and choose **Data store** as the tool type in the drop-down menu.

20. Under **Data Stores**, click on **Add Data Stores**, choose **Australian Red Cross Lifeblood Unstructured**, then click **Confirm** and finally, click **Save**.

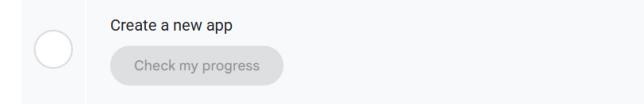
The screenshot shows the 'Create Tool' interface. In the 'Available tools' section, the tool 'Blood_donation_tool' is selected. In the 'Data stores' section, the data store 'Australian Red Cross Lifeblood Unstructured' is listed with its creation date as Mar 16, 2025.

Under **Available tools** choose the tool called **Blood_donation_tool** and click **Save**.

Congratulations! You have finished building your knowledge-powered app that's ready to help potential donors, so take a moment to celebrate!

But there's still more work to do to make the agent accessible to your users. In the next section, you'll use a knowledge handler to enable conversations between the agent and end-users about eligibility requirements.

Click **Check my progress** to verify the objectives.



Task 3. Configure the agent to answer blood eligibility FAQs

Provide the date store prompt

While the document collection process is running in the background, let's give the agent a brand by editing the data store prompt.

1. In the Conversational Agents console and from within your agent, click **Agent settings** (top right corner of the page).



2. Go to the **Generative AI** tab.



3. Click on **General** tab, and set the filters as below.

Filters	Values
Hate Speech	Block few
Dangerous content	Block few(default)
Sexually explicit content	Block few
Harassment	Block few

The screenshot shows the 'Generative AI' tab selected in the top navigation bar. Under the 'Safety filters' section, there are four dropdown menus for different sensitivity levels: 'Safe speech sensitivity level' (set to 'Block few'), 'Dangerous content sensitivity level' (set to 'Block few (default)'), 'Sexually explicit content sensitivity level' (set to 'Block few'), and 'Harmful content sensitivity level' (set to 'Block few'). Each dropdown has a 'Learn more' link below it.

4. Click on **Data store** tab.

The screenshot shows the 'Data store' tab selected in the top navigation bar. Below it, the 'General' tab is also visible. At the top, there are 'Settings', 'Save', and 'Cancel' buttons. The main area contains sections for 'Data store prompt' (with fields for 'Agent name', 'Company name', 'Company description', and 'Agent intent'), 'Data store model selection' (set to 'Default'), and tabs for 'General', 'Generative AI', 'Deterministic Flows', 'Connectivity', 'Speech and IVR', 'UI settings', and 'Security'.

IS **Donate**, and you are a neipuri and poite chatbot at **Save a life**, a fictitious organization. Your task is to assist humans with eligibility information.

The screenshot shows the 'Data store prompt' configuration page. It includes a 'Data store prompt' section with fields for 'Agent name' (set to 'Donate'), 'Company name' (set to 'Save a life'), 'Company description' (set to 'a fictitious organization'), and 'Agent intent' (set to 'humans with eligibility information'). An example message is provided: 'Your name is the ACME Virtual Assistant, and you are a helpful and polite AI Assistant of ACME Co., a fictional e-commerce site. Your task is to assist humans with the company website.' Below this, there's a 'Your prompt:' field and a note about generating responses. At the bottom, there are 'Data store model selection' (set to 'Default') and 'Save' and 'Cancel' buttons.

5. Click the **Save** button located at the top of the tab.

The screenshot shows a 'Generate the data store prompt' interface. It features a large text input field containing the generated prompt, a 'Check my progress' button, and a 'Generate' button.

Enable generative fallback for the Default Start Flow's no-match event

6. Switch to the **Flows** tab on the top-left sidebar of the Conversational Agents console (below of **Playbooks**) and open the **Start Page**.

7. Click the **sys.no-match-default** event handler. Unless the box is already checked, enable the generative fallback feature and click **Save**.

The screenshot shows the 'Event handler' configuration for the 'Default Start Flow'. It lists two event handlers: 'sys.no-match-default' and 'sys.no-input-default'. The 'sys.no-match-default' handler is highlighted. To its right, there's a 'Parameter presets' section with a note about defining response types. A checkbox for 'Enable generative fallback' is checked, with a descriptive note below it. At the top, there are 'Event handler', 'Save', and 'Cancel' buttons.

Review the Fulfillment Agent Response

8. On the **Start Page** click **Default Welcome Intent**.

The screenshot shows the 'Start Page' interface. It has a header with 'Start Page' and a close button ('X'). Below it is a 'Routes' section with a '+' button. The main area is currently empty, indicating no routes have been defined yet.

Event handlers



sys.no-match-default

sys.no-input-default

9. Scroll down to **Agent Responses** under **Fulfillment**. A fulfillment is the agent response to the end user. Conversational Agents has pre populated **Agent dialogue** with the parameter **Hi! How are you doing?.**

Agent responses

Responses using custom payloads. You can also use if else statements to provide conditional responses using code. [Learn more](#)

Agent dialogue



Hi! How are you doing?

Hello! How can I help you?

Good day! What can I do for you today?

Greetings! How can I assist?

Enter agent dialogue

Add

Task 4. Test the agent

Wait until the documents are available and ready for use by your agent to check out how good the answers are. You can check if the documents are available by going to the [Vertex AI Agent Builder console](#) and clicking on the [view](#) link under **Connected data stores** beside of the Blood Donation Agent app and then clicking on Australian Red Cross Lifeblood Unstructured.

Australian Red Cross Lifeblood Unstructured

Your agent can use the content in this datastore to generate responses.

Data store ID	australian-red-cross-lifeb_1704381998968
Type	Unstructured data

Last document import Jan 4, 2024, 10:26:43 AM
[VIEW DETAILS](#)

If you are not in the Conversational Agents console, then from the Vertex AI Agent Builder console, click the name of your app, which will redirect you to the Conversational Agents console.

1. In the Conversational Agents console within your agent, click **Toggle Simulator** icon to open the Simulator.



[Toggle Simulator](#)

- How can I book an appointment?

- What age do I need to be to donate?
- I've just come back from a trip to Africa. Can I donate?
- Can pregnant women donate?

3. Lastly, let's try and challenge the agent with a question totally unrelated to blood donation. For example:

`What's the weather like in Melbourne?`

The agent should respond something like: `I'm sorry, I can't provide weather information.`

This answer has AI generated content in it and derives from the text prompt that Conversational Agents has created starting from the knowledge connector

This text prompt contains the company name, the agent name, and most importantly what is in its scope which is used by Conversational Agents to generate the agent response.

Well done! So far you are using the data store to assist people with frequently asked questions related to blood donation. In the next part of the lab, we will look at how to bind a generator text prompt to the same content to make informed decisions.

Task 5. Set up the agent for the eligibility quiz

There are strict requirements donors must meet such as age, weight, existing conditions, recent travels, etc. For the scope of this lab, we will only consider age and weight. A generator will use Google's large language models (LLMs) to dynamically make an informed decision based on the context of the conversation and the knowledge base.

Configure new routes and parameters

1. Switch to the **Flows** tab on the top-left sidebar of the Conversational Agents console (below of **Playbooks**) and open the **Start Page** and then click **Default Welcome Intent**.
2. Scroll down to the **Fulfillment** section and remove the current agent responses in the **Agent dialogue** field and insert the following response:

The screenshot shows the left sidebar with 'Routes' and 'Event handlers' sections. Under 'Event handlers', there are entries for 'sys.no-match-default' and 'sys.no-input-default'. A 'Add state handler' button is also visible. On the right, the 'Fulfillment' section is open, showing a 'Parameter presets' dropdown and a 'Generators' dropdown. Below these is the 'Agent responses' section, which includes an 'Agent dialogue' area with a message: 'Would you like to take the eligibility quiz to find out if you can donate blood, and start changing lives?'.

3. Scroll down to the **Transition** field and select **Page > + new page** and set page name as **User Blood Donation Decision** and click on **Save** button.

4. Navigate to the **Manage** tab to create the two intents with the below configurations, click **+ Create**

Display Name	Property
confirmation.yes	Training phrases: "Yes", "yeah", "yes please"
confirmation.no	Training phrases: "No"

5. Navigate to **Build** tab and click on the **User Blood Donation Decision** page.

- To create the routes, click on **+** icon next to **Routes**.
- In the **Intent** field, choose **confirmation.yes** from dropdown.

page name as **Eligibility Quiz** and click on **Save** button.

- Click on **+** icon next to **Routes**.
- In the **Intent** field, select **confirmation.no** from the dropdown menu. Then, scroll down to the **Fulfillment > Agent responses** section, click **+ Add dialogue response**, choose **agent dialogue** and insert the response as **Thanks, Have a nice day!** then click on **Save**.

Create and configure the eligibility generator

The [generator feature](#) is a Conversational Agents feature that allows developers to use Google's latest generative large language models (LLMs) during Conversational Agents fulfillment. Generators to generate agent responses at runtime. A generator can handle generic responses that involve general knowledge from a large textual dataset it was trained on or context from the conversation.

(such as age and weight) with the eligibility requirements to determine whether the user can donate.

6. On the Conversational Agents console go to the **Manage** tab and select **Generators** and click **Create new**.

The screenshot shows the 'Generators' section under the 'Manage' tab. The sidebar has sections for 'MESSAGES', 'TEST & FEEDBACK', and 'TESTING & DEPLOYMENT'. The main area displays a message: 'No generator is created yet. Generators allow you to use generative AI models to generate generic responses or text that can be used during fulfillment.' A 'Create new' button is visible.

7. Next, provide **Blood Donation Eligibility** as the display name and write the

Check the users eligibility against the following criteria:
the minimum age is 18 and the maximum age is 75. Weight
should be above 50 Kg. The user age and weight is \$last-
user-utterance. Be nice and tell the user if they are
eligible to donate (also tell them why not in case)

8. Leave the default model quality control settings. Then click **Save** to create the generator.

The text prompt is sent to the generative model during fulfillment at runtime. It should be a clear question or request in order for the model to generate a satisfactory response. You can use special built-in generator prompt placeholders in your text prompt:

- `$conversation`: The conversation between the agent and the user, excluding the very last user utterance.

The text prompt you have configured expects the user to provide age and weight in once conversational turn (the last-userutterance).

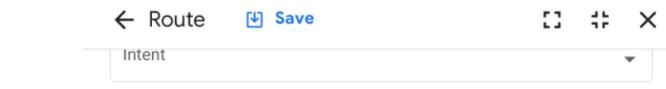
Use the generator in fulfillment and configure all the required parameters

9. Next, navigate to **Build** tab and click on **Eligibility Quiz Page**.

- To add Parameters, click on + icon next to **Parameters**.
- Enter display name as `age-weight`.
- Select entity type as `@sys.any`.
- Scroll down to **Initial Prompt Fulfillment > Agent responses** and in

10. On **Eligibility Quiz** page, click on + icon next to **Routes**.

- Select the `Match AT LEAST ONE rule (OR)` option under **Condition rules** and enter the condition requirement `$page.params.status = "FINAL"`.



Condition

A conditional trigger determines how the route will occur. For example,

Condition rules

- Match **AT LEAST ONE** rule (OR)

- Match **EVERY** rule (AND)

- Customize expression

Parameter: `$page.params.status` Operand: `=` Value: `"FINAL"`

[Add Rule \(OR\)](#)

- Scroll down to **Fulfillment > Generators**, click on **Add Generator**, and select **Blood Donation Eligibility** as the generator. Then, replace the value in the **Output Parameter** field with

`$request.generative_eligibility_outcome` that will contain the

Generators

Generators allow you to use generative AI models to generate dynamic responses or text that can be used during fulfillment. [Learn more](#)



Blood Donation Eligibility

Edit

Text prompt ⓘ

Check the users eligibility against the following criteria: the minimum age is 18 and the maximum age is 75. Weight should be at least 50 Kg. The user's age and weight are \$last-user-utterance. Be kind and tell the user if they're eligible to donate

Output parameter

[Redacted]

- In Fulfillment > Agent responses > Agent dialogue field add value as `$request.generative.eligibility-outcome` and then click **Save**.

Task 6. Retest your agent

- Click on **Toggle Simulator** icon, available at top right corner.

Toggle Simulator

- Click the **Reset conversation** icon until the **Preview: Default Start Flow** page appears, as shown below.

Preview: Default Start Flow

Reset conversation

Preview: Default Start Flow

Send a message to see how your agent responds

Default Start Flow

Start page

- In the **Start Resource** section of the **Preview: Default Start Flow** page, select **Default Start Flow** and test your agent.

Preview: Default Start Flow

Send a message to see how your agent responds

Default Start Flow

Start page

Start Resource

Default Start Flow

- In the **Toggle Simulator**, start a new conversation with the agent by entering **Hi**, and then provide the response to the agent's questions as shown below.

Preview: Default Start Flow

Hi

Would you like to take the eligibility quiz to find out if you can

A screenshot of a chatbot interface. The user asks: "donate blood, and start changing lives?" The bot responds with "Yes". The user then asks "What is your age and weight?", to which the bot replies "I am 25 and my weight is 70 Kg". The bot concludes with "Congratulations! Based on your age (25) and weight (70 kg),"

5. Then check the eligibility check fails when one or both the requirements are not met.

A screenshot of a chatbot interface titled "Preview: Default Start Flow". The user says "Hi". The bot asks "Would you like to take the eligibility quiz to find out if you can donate blood, and start changing lives?". The user replies "Yes". The bot then says "I am 19 years old and my weight is 47 Kg". It responds with a message: "Hello! Thank you for wanting to donate. While we appreciate your interest, it seems you don't quite meet all the eligibility requirements at this time. You're 19 years old, which is within the acceptable age range (18-75), but your weight of 47 kg is below the minimum of 50 kg required for donation. Please come back when you meet the weight requirement! We hope to see you then."

6. Great, the generator works as expected! Or does it? What happens if the user provides the age but not the weight (or the other way around)?

A screenshot of a chatbot interface titled "Preview: Default Start Flow". The user says "Hi". The bot asks "What is your age and weight?". The user replies "I am 23". The bot responds with a message: "Hello! Based on the information you provided (age 23), you are eligible to donate! Your age is well within the acceptable range (18-75). We'll need to check your weight to confirm your full eligibility, though. Please provide your weight so we can complete the assessment."

Task 7: Generator prompt tuning

Collecting age and weight in one ago doesn't seem to work unless both age and weight are provided. We should instead create a form that collects both values as entity parameters. To make the prompt contextual of all the eligibility requirements (such as the age and the weight) we can use placeholders by adding a \$ before the word. We will later associate these generator prompt placeholders with session parameters in fulfillment and they will be replaced by the session parameter values during execution.

1. Open the **Eligibility Quiz** page, click on **Parameters**, remove the parameter `age-weight` and add two separate form parameters: one for `weight` and one for `age`. Pick `@sys.number-integer` as the entity type and mark the parameters

required. Provide the initial prompt fulfillments such as `What is your correct weight?` for the `weight` parameter and `How old are you?` for the `age` parameter. Save all the changes.

Once you've updated the text prompt of the generator, click **Save**.

against the following criteria: the minimum age is 18 and the maximum age is 75. The weight must be at least 50 kg. The user is \$age years old and weighs \$weight Kg. Craft an email and politely explain to the user if they're eligible to donate and if not why.

Notice that we haven't just made the text prompt contextual of the age and weight form parameters, we have also changed the last sentence to be able to generate a formal email to the user which contains the official outcome of the eligibility quiz.

Craft an email and politely explain to the user if they're eligible to donate and if not why.

4. Click **Save**.

5. Return to the **Build** tab. On the **Eligibility Quiz** page, select the route and expand the **Generators** section of the **Fulfillment** pane. Then, click **Add generator** and select the **Blood Donation Eligibility** generator. After selecting the generator you need to associate the new prompt placeholders with the respective session parameters. Moreover, you need to re-set the `input` and `output` parameters as below.

- `age : $session.params.age`

- `weight : $session.params.weight`

- Output parameter : \$request.generative.eligibility-outcome

responses or text that can be used during fulfillment. [Learn more](#)

Generator

Generator
Blood Donation Eligibility
Edit

Text prompt ⓘ

Check the users eligibility against the following criteria: the minimum age is 18 and the maximum age is 75. The weight must be at least 50 kg. The user is \$age years old and weighs \$weight Kg. Be kind and tell the user if they're eligible to donate.

Input parameters

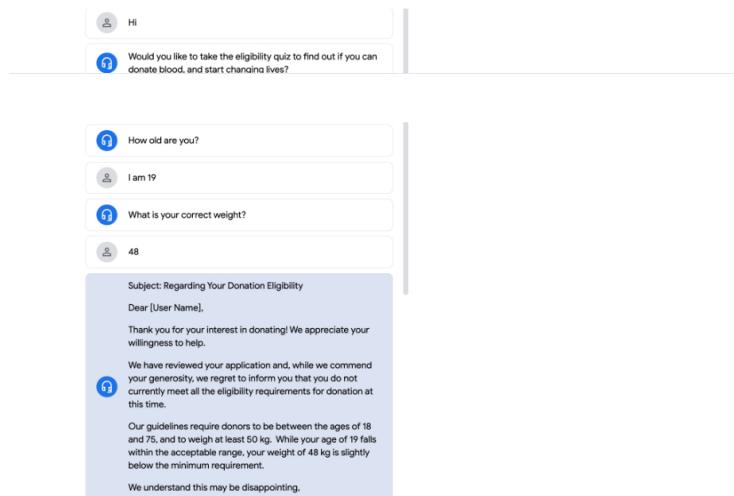
Placeholder	Parameter
age	\$session.params.age
weight	\$session.params.weight

Output parameter

\$request.generative.eligibility-outcome

6. Click on **Save**.

7. Retest the agent again. The eligibility check takes now into account both age and weight and the wording has changed from a conversational tone to a more polite response that is ready to be sent out without any potential human in the loop.



Congratulations!

Today we've investigated generators in the context of eligibility quizzes. You have seen that generators use LLMs to generate agent responses and when powered by a knowledge base they can also make well-informed decisions. Surely there are many other use cases that can be implemented leveraging generators and data stores and we can't wait to get to know them!

[Learn more](#)

Continue learning about Agent Builder AI and generative AI with these guides and

resources:

- [Introduction of Vertex AI Agent Builder](#)
- [Create and use a data store](#)
- [Documentation for Vertex AI Agent Builder](#)
- [Generative AI in Google Cloud](#)

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated March 12, 2025

Lab Last Tested March 12, 2025

Copyright 2023 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.

Conversational