

Google Assistant: Build an Application with Dialogflow and Cloud Functions

GSP174



Google Cloud Self-Paced Labs

Overview

[Google Assistant](#) is a personal voice assistant that offers a host of actions and integrations. From making appointments and setting reminders, to ordering coffee and playing music, the 1 million+ actions available suit a wide range of voice command tasks. Google Assistant is offered on Android and iOS, but it can even be integrated with other devices like smartwatches, Google Homes, and Android TVs.

[Actions](#) is the central platform for developing Google Assistant applications. The Actions platform integrates with human-computer interaction suites, which simplifies conversational app development. The most widely used suite is [Dialogflow](#), which uses an underlying machine learning (ML) and natural language understanding (NLU) schema to build rich Assistant applications. The Actions platform also integrates with [Cloud Functions](#), which lets you run backend fulfillment code in response to events triggered by Dialogflow requests.

In this lab, you will get hands-on practice with the Actions platform, the Dialogflow suite, and Cloud Functions by building a "Silly Name Maker" application, which returns a user with a silly name after they have entered in a lucky number and favorite color. You will build a Dialogflow agent that intelligently parses user input for specific information. The agent will be supplemented with a webhook, which will trigger a Cloud Function that handles fulfillment logic and returns your user with their silly name.

What you will learn

In this lab, you will learn how to:

- Create an Actions project and build an Action.
- Create a Dialogflow agent and configure the default welcome intent.
- Build a custom intent with entities.
- Initialize a Cloud Function.
- Add fulfillment logic and packages to your Cloud Function.
- Add a webhook to your Action.
- Test your Assistant application with the Actions simulator on expected and unexpected conversational paths.
- **Optional:** test your Assistant application on a Google Home device.

Prerequisites

This is a **fundamental level** lab. Familiarity with the Actions Console and the Qwiklabs platform is expected. If you need to get up to speed with the lab's requirements, please complete one of the following Qwiklabs:

- [A Tour of Qwiklabs and the Google Cloud](#)
- [Google Assistant: Qwik Start - Dialogflow](#)

Since this lab works with the Actions simulator, having a pair of headphones or turning the volume up on your computer is recommended. If you want to test your Assistant application on a Google Home, keep your device handy.

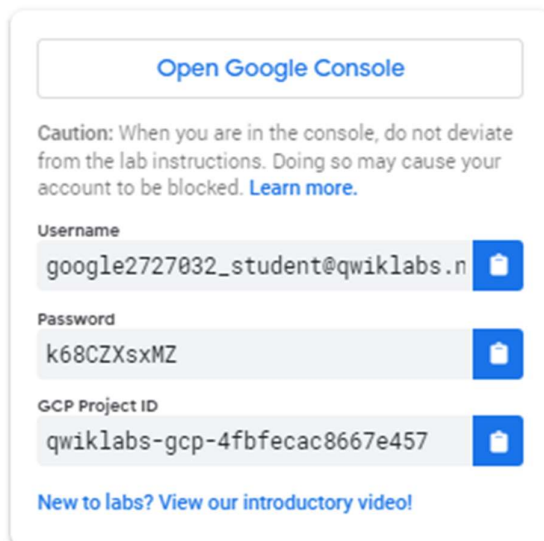
Once you're ready, scroll down and follow the steps below to set up your lab environment.

Setup

Cloud Console

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



[Open Google Console](#)

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)

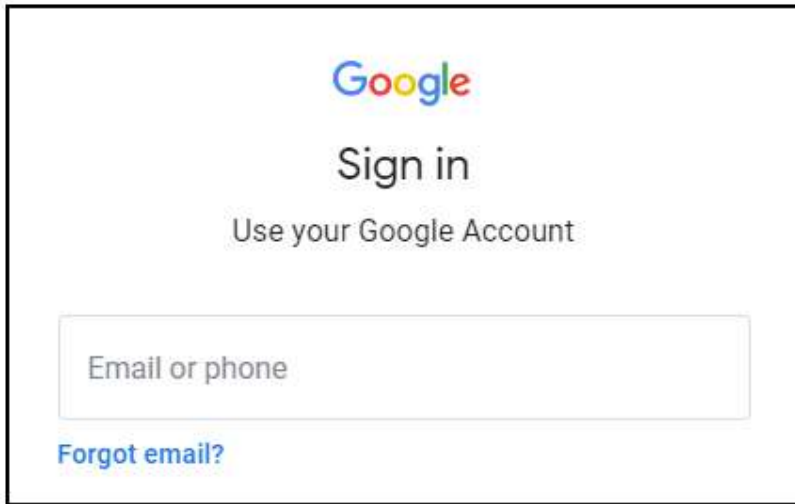
Username
google2727032_student@qwiklabs.n

Password
k68CZXsxMZ

GCP Project ID
qwiklabs-gcp-4fbfecac8667e457

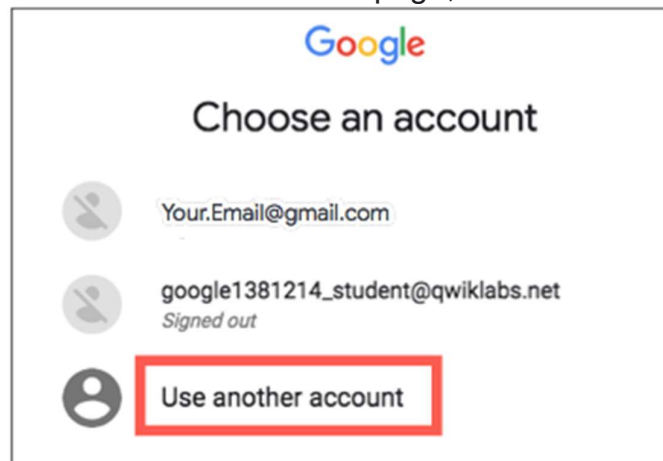
[New to labs? View our introductory video!](#)

2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Tip: Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another**



Account.

3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

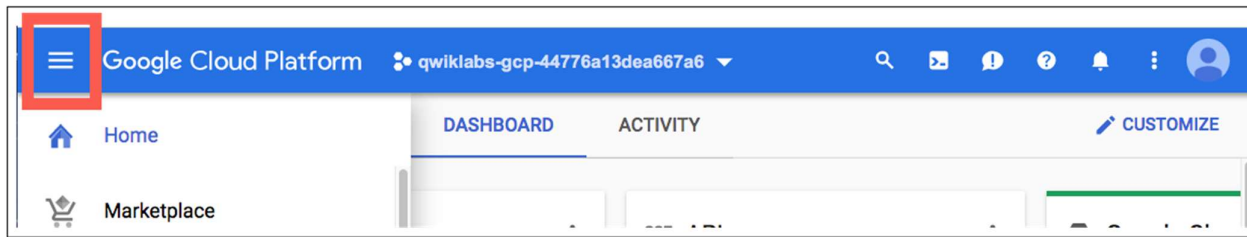
Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



Create an Actions project

Regardless of the Assistant application you're building, you will always have to create an Actions project so your app has an underlying organizational unit.

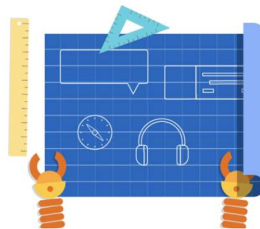
Open the [Actions on Google Developer Console](#) in a new tab. Sign in with your Qwiklabs credentials if prompted. You should be looking at a clean Actions console that resembles the following:

Welcome to Actions on Google

Actions on Google is the platform for developers to extend the Google Assistant. Join this emerging ecosystem by developing actions to engage users on Google Home, Pixel, and many other surfaces where the Google Assistant will be available.

[Documentation](#) [Sample code](#) [API reference](#) [Support](#)

Your projects



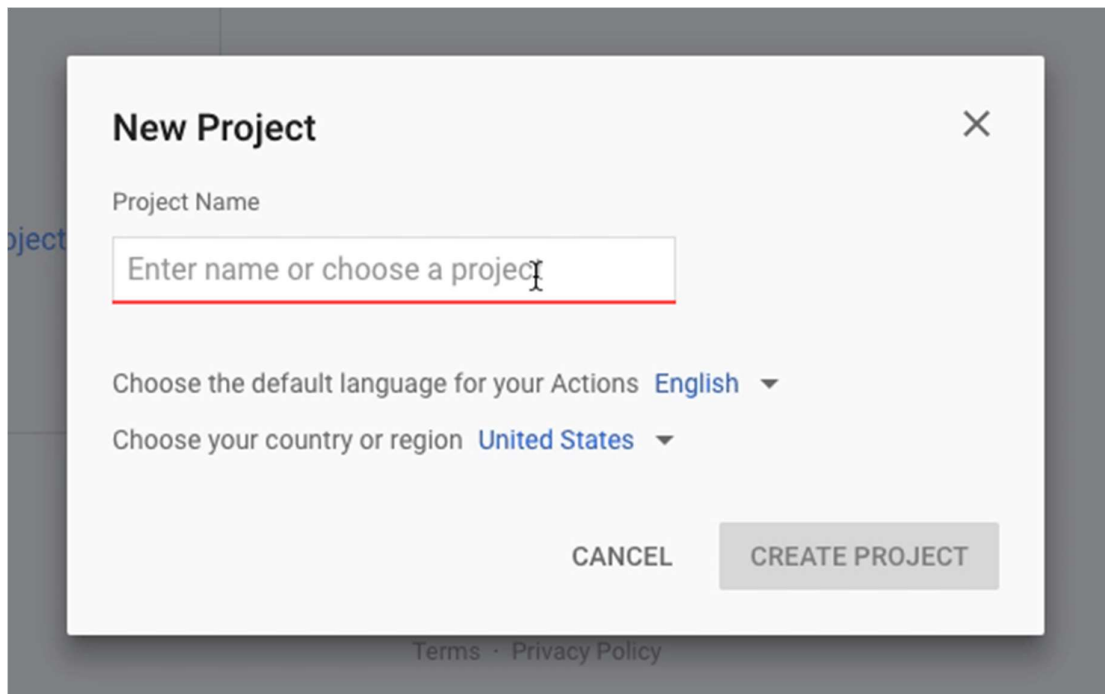
You don't have any projects yet. Click below to get started.

New Project

[Terms](#) · [Privacy Policy](#)

Click **New Project** and agree to Actions on Google's terms of service when prompted by clicking **Agree and continue**.

Click into the `Project Name` field and select your Qwiklabs Google Cloud project ID from the dropdown. Then click **Import project**:

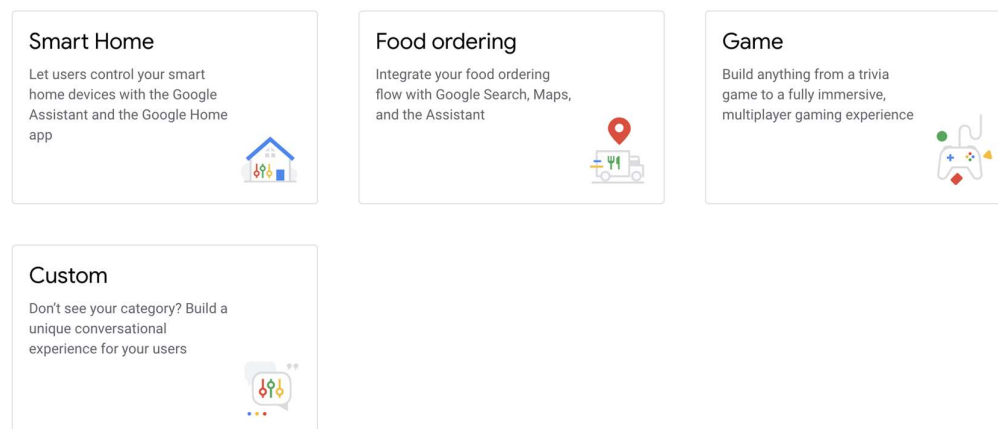


The screenshot shows a 'New Project' modal dialog. At the top, it says 'New Project' with a close button (X). Below is a 'Project Name' section with a text input field containing the placeholder 'Enter name or choose a project'. Underneath the input field are two dropdown menus: 'Choose the default language for your Actions' set to 'English' and 'Choose your country or region' set to 'United States'. At the bottom right are two buttons: 'CANCEL' and 'CREATE PROJECT'. At the bottom center, there are links for 'Terms' and 'Privacy Policy'.

Soon after you will be presented with a welcome page that resembles the following:

What kind of Action do you want to build?

Select the category that best fits the type of experience you want to build for the Google Assistant.



The screenshot shows a selection screen with four categories, each with a description and an icon:

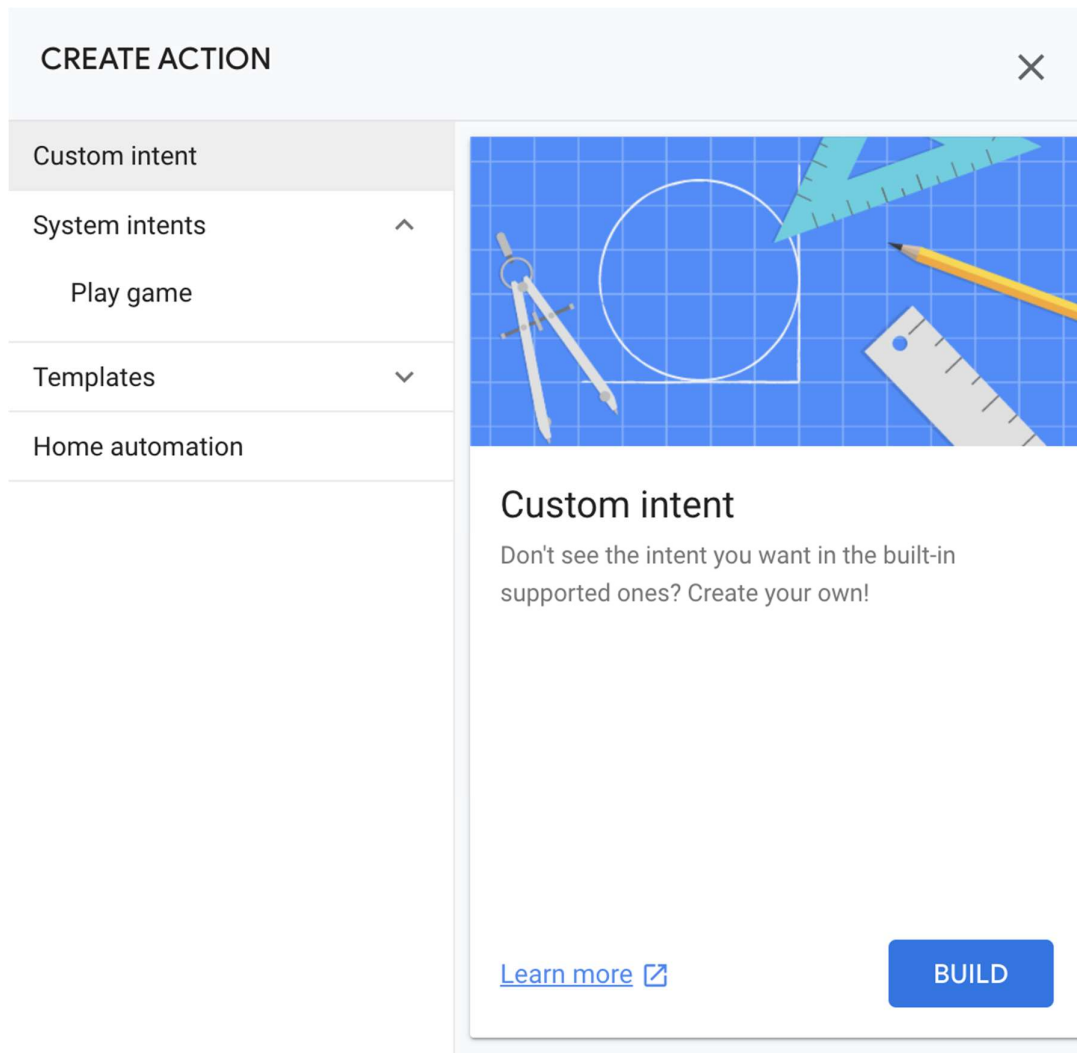
- Smart Home**: Let users control your smart home devices with the Google Assistant and the Google Home app. Icon: A house with smart home symbols.
- Food ordering**: Integrate your food ordering flow with Google Search, Maps, and the Assistant. Icon: A food truck with a location pin.
- Game**: Build anything from a trivia game to a fully immersive, multiplayer gaming experience. Icon: A video game controller.
- Custom**: Don't see your category? Build a unique conversational experience for your users. Icon: A speech bubble with a person icon.

Now click **Actions Console** in the top left corner to return to the homepage. Then click on the project you just created (title has your Project ID as the name.)

Build an Action

An [action](#) is an interaction you build for the Google Assistant. An action supports a specific [intent](#) (a goal or task that users want to accomplish), which is carried out by a corresponding [fulfillment](#) (logic that handles an intent and carries out the corresponding Action.) You will now build an Action that supports silly name generation.

Click on your project name. Then from the center menu click **Build your Action > Add Action(s) > Get Started**. Then select **Custom Intent > BUILD**:



This will take you to the Dialogflow console. Select your Qwiklabs account and click **Allow** when Dialogflow prompts you for permission to access your Google Account.

When you land on the Dialogflow account settings page, check the box next to **Yes, I have read and accept the agreement** and click **Accept**.

If you are brought to the following Dialogflow agent creation page, click **CREATE**:

qwiklabs-gcp-8aaed5810f687c99

CREATE

DEFAULT LANGUAGE ?

English — en

Primary language for your agent. Other languages can be added later.

DEFAULT TIME ZONE

(GMT-8:00) America/Los_Angeles

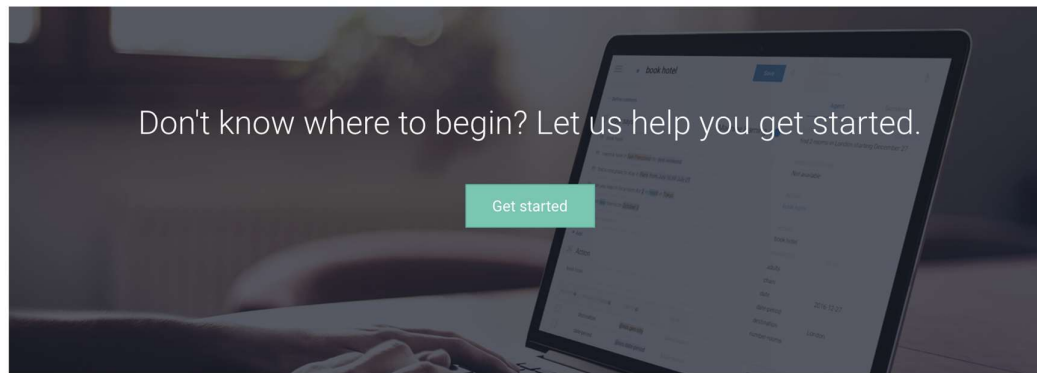
Date and time requests are resolved using this timezone.

GOOGLE PROJECT

Agent will be linked with [qwiklabs-gcp-8aaed5810f687c99](#) Google Project

If you are brought to this page instead:

 Welcome to Dialogflow!



Now it's time to create your first agent.

CREATE AGENT

Close the Dialogflow agent creation tab. You will return to the Actions Console.

Click **Get Started** > **Custom Intent** > **BUILD**.

Select your Qwiklabs account and click **Allow** when Dialogflow prompts you for permission to access your Google Account.

Now click **CREATE**:

qwiklabs-gcp-8aaed5810f687c99

CREATE

DEFAULT LANGUAGE ?

English — en

Primary language for your agent. Other languages can be added later.

DEFAULT TIME ZONE

(GMT-8:00) America/Los_Angeles

Date and time requests are resolved using this timezone.

GOOGLE PROJECT

Agent will be linked with [qwiklabs-gcp-8aaed5810f687c99](#) Google Project

An [agent](#) is an organizational unit that collects information needed to complete a user's request, which it then forwards to a service that provides fulfillment logic. You will now build the basic framework for fulfillment logic. This will be handled (later) by a Cloud Function, which will return a response with a user's silly name.

Test Completed Task

Click **Check my progress** to verify your performed task.

Click **Fulfillment** from the left-hand menu. Move the slider for **Webhook** to the right, setting it to **Enabled**.

Now enter the temporary URL <https://google.com> in for the URL field. You will update this URL when you build your Cloud Function. Your page should resemble the following:

⚡ Fulfillment

Webhook

ENABLED 

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched by intents with webhook enabled. Be sure that your web service meets all the [webhook requirements](#) specific to the API version enabled in this agent.

URL*

BASIC AUTH

Scroll down and click **Save** in the bottom right corner. Then click **Intents** from the left hand menu and select **Default Welcome Intent**:



Intents

Search intents



Default Fallback Intent



Default Welcome Intent

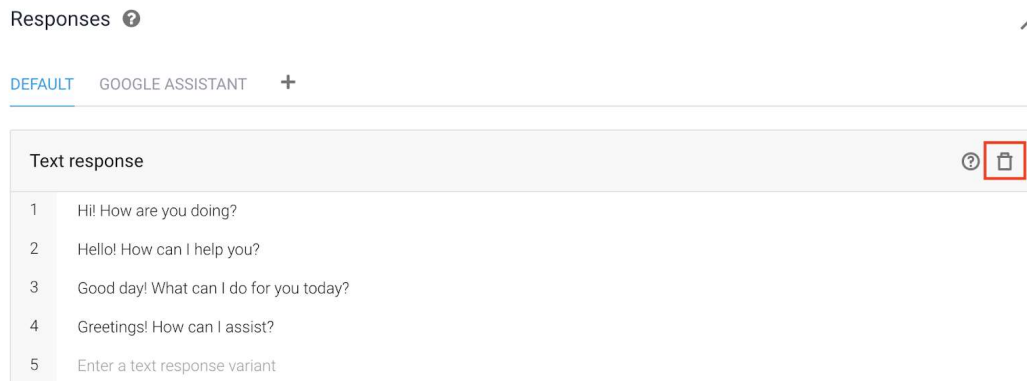
You will now build the main entry point into your application by configuring the default welcome intent.

Configure the default welcome intent

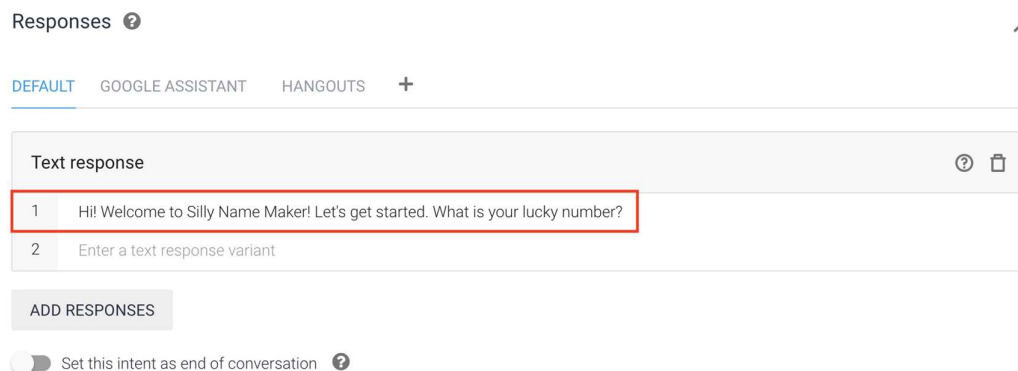
When you created the Dialogflow agent, a **Default Welcome intent** is automatically created. This intent represents the main entry point into your conversation and the main *action* of your app. Your app must have a Default Welcome Intent defined, so that Actions on Google knows how to invoke your app. See [Invocation and Discovery](#) for more information on how these invocation models work.

Make sure that you are in the Default Welcome Intent. Here are some things to notice about the default settings:

- The **Events** section of the intent specifies a Welcome event, which signifies that this intent is the default entry point into your app. The Google Assistant uses this event to trigger your app when users invoke your app by name, such as with *"Ok Google, talk to Silly Name Maker"*.
- In the **Responses** section, there are default, static text responses. The **Default Welcome Intent** contains pre-populated responses that you should remove to add your own. Now that you know the Default Welcome Intent's purpose, you will modify some fields. Scroll down to the **Responses** section and click the **trash icon** to scrap all of the default text responses:



Now click on **ADD RESPONSES > Text response**, and type in the following: *Hi! Welcome to Silly Name Maker! Let's get started. What is your lucky number?*



Now scroll up and click **Save** in the top-right corner. Now when users invoke your app, they know that they are entering your app's experience and what they should say next.

Test Completed Task

Click **Check my progress** to verify your performed task.

In general, the app's responses should guide users on what to say next and should stay within your conversations grammar. This will be mapped out further in the following section.

Design the conversation

Before you begin building out your Assistant application's language path, it's always a good idea to map out a conversation by writing sample dialogs. For the Silly Name Maker app, we've provided a couple of simple dialogs, which are described below:

Expected path

- **App:** Hi! Welcome to Silly Name Maker! Let's get started. What is your lucky number?
- **User:** 5
- **App:** What is your favorite color?
- **User:** Yellow
- **App:** Alright, your silly name is Yellow 5! I hope you like it. See you next time.

Unexpected path

- **App:** Hi! Welcome to Silly Name Maker! Let's get started. What is your lucky number?
- **User:** Dog
- **App:** Hmm that doesn't sound like a number to me. What's your lucky number again?
- **User:** 22
- **App:** What is your favorite color?
- **User:** Blue
- **App:** Alright, your silly name is Blue 22! I hope you like it. See you next time.

Conversational design tips: This example shows only a couple of the dialogs that you could write for your own apps. See this [Design Walkthrough](https://developers.google.com/actions/design/walkthrough#write_dialogs) Design for more examples of dialogs.

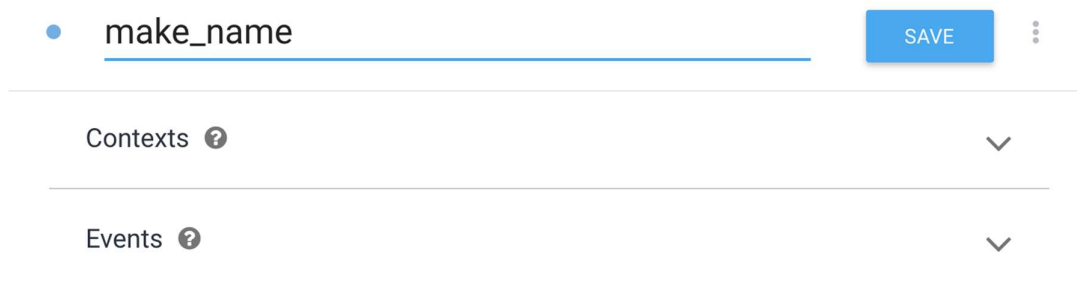
Now that you have a better understanding of the type of application you'll be building, you will develop your custom intent to parse and extract colors and numbers from user input.

Configure a custom intent

Dialogflow intents define your conversation's grammar and what tasks to carry out when users make specific requests. Since Dialogflow includes a natural language understanding (NLU) engine, you don't need to be exhaustive when defining phrases. Dialogflow automatically expands on the phrases you provide and understands many more variations of them.

You will now create an intent that defines how users need to provide their lucky number and favorite color to generate a silly name.

From the left hand menu, click on the **+** icon by the **Intents** menu item. For the **Intent name** field, enter `make_name`:



The screenshot shows the Dialogflow 'Intents' configuration page. At the top, the intent name 'make_name' is entered in a text field, followed by a blue 'SAVE' button and a vertical ellipsis menu icon. Below this, there are two expandable sections: 'Contexts' and 'Events', each with a question mark icon and a downward arrow.

In the **Training phrases** field click **Add Training Phrases**. Then in the Add user expression field enter in `My lucky number is 23`. Highlight the number 23 and assign it the `@sys.number` "entity" and hit **Enter**.

Your actions and parameters section should resemble the following:



Add user expression

My lucky number is 23

PARAMETER NAME	ENTITY	RESOLVED VALUE	
number	@sys.number	23	

Entities extract specific data from user expressions and store them in accessible variables. By assigning this portion of the user says to be an entity, Dialogflow can extract parameters from the user input, validate the parameter and provide it to your fulfillment as a variable.

Without assigning entities, you would have to parse the input yourself and find the parameters you need.

You will now add some more user expressions in the **Training phrases** section, and assign any number within those phrases the `@sys.number` entity (if it isn't automatically set.) Add the following training phrases by entering them in the **Add user expression** field:

- 23
- *The luckiest number I have is 12*
- *My lucky number is 18*

Your Training phrases section should now resemble the following — note the highlighted numbers, which align with the `@sys.number` entity:



Add user expression

My lucky number is 18

The luckiest number I have is 12.

23

My lucky number is 23

Now that your training phrases have been added, scroll down and expand the **Actions and parameters** section by clicking **MANAGE PARAMETERS AND ACTION**.

Check the **REQUIRED** checkbox for the number parameter. This tells Dialogflow to not trigger the intent until the parameter is properly provided by the user.

Now click on **Define prompts** for the number parameter (right-hand side) and provide a re-prompt phrase. Type in *What's your lucky number?* in the prompt field and then click **Close**:

Prompts for "number"

NAME	ENTITY	VALUE
number	@sys.number	\$number

PROMPTS

1

What's your lucky number?

2

Enter a prompt variant

CLOSE

This phrase will be spoken to the user repeatedly until Dialogflow detects a number in the user input.

In the Action and parameters section, add a parameter with the following information:

- **Required** - Select the checkbox
- **Parameter name** - color
- **Entity** - @sys.color
- **Value** - \$color
- **Prompts** - What's your favorite color?

This additional parameter uses Dialogflow's built-in "slot-filling" feature, which allows you to obtain additional input parameters from the user without having to create an intent for each one and without having to make the user speak all the required input in one phrase.

Slot filling also lets you set parameters as being required so that your agent doesn't process the input until the user provides all required parameters. Your actions and parameters should now resemble the following:

Action and parameters

Enter action name

REQUIRED ?	PARAMETER NAME ?	ENTITY ?	VALUE	IS LIST ?	PROMPTS ?
<input checked="" type="checkbox"/>	number	@sys.number	\$number	<input type="checkbox"/>	What's your luc...
<input checked="" type="checkbox"/>	color	@sys.color	\$color	<input type="checkbox"/>	What's your fav...

+ New parameter

Now scroll to the top of the page and click **Save**.

Now scroll down and expand the **Fulfillment** section and click **Enable fulfillment**. Then click the **Enable webhook call for this intent** slider:

Fulfillment ?

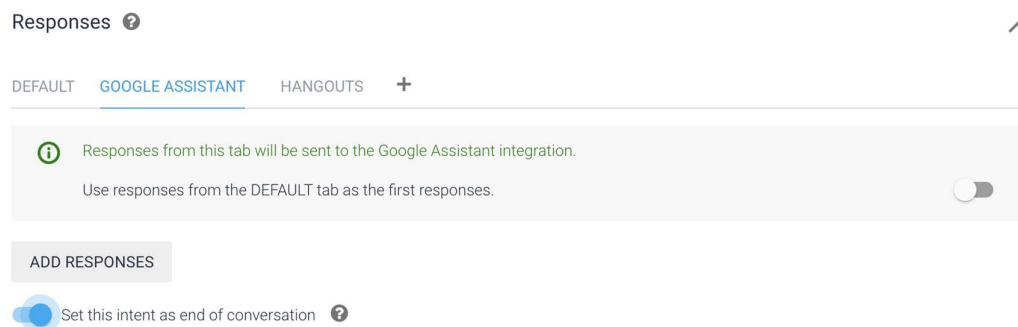
☒ Enable webhook call for this intent

☐ Enable webhook call for slot filling

This tells Dialogflow to call your fulfillment to generate a response to the user instead of using Dialogflow's response feature.

In the **Responses** section right above Fulfillment, click on **+** icon and select **Google Assistant**.

Move the toggle for **Set this intent as end of conversation**. This tells Dialogflow to relinquish control back to the Google Assistant after your fulfillment returns a response to the user.



Then scroll to the top of the intent and click **SAVE** once more to save the entire intent.

Test Completed Task

Click **Check my progress** to verify your performed task.

You have now successfully declared your conversation's grammar with Dialogflow intents. You have also used Dialogflow's built-in response feature to return a static response to the user when they invoke your app. You also created an intent that uses fulfillment to return a response, which you will now implement with a Cloud Function.

Initialize and configure a Cloud Function

You will now build a Cloud Function to handle your fulfillment logic. Your fulfillment takes the extracted user input that Dialogflow parsed and responds with the user's silly name. **Return to the Cloud Console for this step.**

Open the **Navigation menu** and select **Cloud Functions**, which is located under the compute header. Then click **Create function**.

This will open a template to create a new Cloud Function. Your page will resemble the following:

Cloud Functions ← Create function

1 Configuration — 2 Code

Basics

Function name *
function-1

Region
us-central1

Trigger

⌕ HTTP

Trigger type
HTTP

URL
https://us-central1-qwiklabs-gcp-01-c4d8b493f7ab.cloudfunctions.net/function-1

Authentication

☐ Allow unauthenticated invocations
Check this if you are creating a public API or website.

☒ Require authentication
Manage authorized users with Cloud IAM.

SAVE CANCEL

VARIABLES, NETWORKING AND ADVANCED SETTINGS

For the Cloud **Function Name** field, enter in `silly-name-maker`.

Then scroll down to the **authentication** section and check the box next to `Allow unauthenticated invocations`:

Basics

Function name *

silly-name-maker



Region

us-central1



Trigger

HTTP

Trigger type

HTTP



URL

https://us-central1-qwiklabs-gcp-01-c4d8b493f7ab.cloudfunctions.net/silly-name-...

Authentication

☒ Allow unauthenticated invocations
Check this if you are creating a public API or website.

☐ Require authentication
Manage authorized users with Cloud IAM.

SAVE

CANCEL

Click **Save** then click **Next**.

Forgetting to do the above will cause your simulation test to fail at the end!

Now scroll down and find the inline editor for `index.js` and `package.json`. Make sure that the `index.js` tab is open. This file defines your fulfillment logic and is used to create and deploy a Cloud Function. Here are some specifics on its basic functioning:

- When Dialogflow intents are triggered, the intent's action name (declared in the action area of the intent) is provided to you in the request to your fulfillment. You use this action name to determine what logic to carry out.
- Within every request to your fulfillment, if Dialogflow parsed parameters from the user input, you can access the parameter by name. Here, you declare the names of the parameters so you can access them later.
- This function fulfills the action by generating a silly name. It's called whenever the `make_name` intent is triggered. It uses the parameters from the user input to generate the name.

Now that you have a better understanding of `index.js`, you will build out the function's fulfillment logic. Remove the boilerplate code from the file. Then copy and paste the following code into `index.js`:

```
'use strict';

const {dialogflow} = require('actions-on-google');
const functions = require('firebase-functions');

const app = dialogflow({debug: true});

app.intent('make_name', (conv, {color, number}) => {
  conv.close(`Alright, your silly name is ${color} ${number}! ` +
    `I hope you like it. See you next time.`);
});

exports.sillyNameMaker = functions.https.onRequest(app);
```

Now open the `package.json` tab. This file declares package dependencies for your fulfillment, including the Actions client library. Replace the contents of the file with the following:

```
{
  "name": "silly-name-maker",
  "description": "Find out your silly name!",
  "version": "0.0.1",
  "author": "Google Inc.",
  "engines": {
    "node": "8"
  },
  "dependencies": {
    "actions-on-google": "^2.0.0",
    "firebase-admin": "^4.2.1",
    "firebase-functions": "1.0.0"
  }
}
```

Once you have those files configured, find the **Entry point** field. Enter `sillyNameMaker` for the value.

Now click the **Deploy** button below. It will take about a minute for your function to be built. When the creation completes, your overview page will resemble the following:

Cloud Functions

Functions

CREATE FUNCTION

REFRESH

DELETE

COPY

Filter functions

<input type="checkbox"/>	Name ↑	Region	Trigger	Runtime	Memory allocated	Executed function	Last deployed	Authentication ?	Actions
<input checked="" type="checkbox"/>	silly-name-maker	us-central1	HTTP	Node.js 10	256 MiB	sillyNameMaker	Nov 7, 2020, 3:46:35 PM	Allow unauthenticated	<div></div>

Now click on the `silly-name-maker` function to get more details about its configuration. Then click on the **Trigger** tab. You will see a trigger URL that resembles the following:

silly-name-maker

Version 1, deployed at Nov 7, 2020, 3:46:35 PM... ▼

METRICS

DETAILS

SOURCE

VARIABLES

TRIGGER

PERMISSIONS

LOGS

TESTING

HTTP

Trigger URL

<https://us-central1-qwiklabs-gcp-01-c4d8b493f7ab.cloudfunctions.net/silly-name-...>

Copy the trigger URL. You will use it as the URL for the Dialogflow webhook, which is configured in the next section.

Test Completed Task

Click **Check my progress** to verify your performed task.


Configure the webhook

Return to the Dialogflow console and click on the **Fulfillment** menu item from the left hand navigation menu.

In the URL field, replace `https://google.com` with the trigger URL you generated in the previous step.

Your webhook should now resemble the following:

Webhook


ENABLED 

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched by intents with webhook enabled. Be sure that your web service meets all the [webhook requirements](#) specific to the API version enabled in this agent.

URL*

BASIC AUTH

HEADERS

 Add header

SMALL TALK

☐ Disable webhook for Smalltalk

Scroll down to the bottom of the page and click **Save** in the lower right corner.

Test your Assistant application with the Actions simulator

Now that you your Cloud Function has been deployed and your webhook has been properly set up, you can preview the app in the Actions simulator.

Check your Google permission settings

In order to test the Silly Name Maker, you need to enable the necessary permissions.

Open a new tab in your browser and visit the [Activity Controls page](#). Sign in with your Qwiklabs credentials if requested.

Ensure that the following permissions are enabled by sliding the toggles to **TURN ON** the following cards:


- **Web & App Activity**


Now **close** the Activity Controls page.


Test the application with the simulator

Return to the Dialogflow console. Then from the left-hand menu, click **Integrations**. Then select **Integration Settings**.

Once you land on the following page, click **TEST**:


 **Google Assistant**
Try out the new [Actions Builder](#) for Google Assistant

 After the next draft submission, changes made in the Dialogflow will no longer impact existing Action versions right away. Instead, you can continue iterating and improving your Action in draft mode and only make it available to users when you're ready
[LEARN MORE](#)

 **Discovery**

Explicit invocation *

Sign in required ?

Default Welcome Intent 

☐



Specify the intent that is triggered when users request the app by name (for example "Ok Google, talk to Personal Chef."). [Learn more.](#)

Implicit invocation


Sign in required ?

Add intent

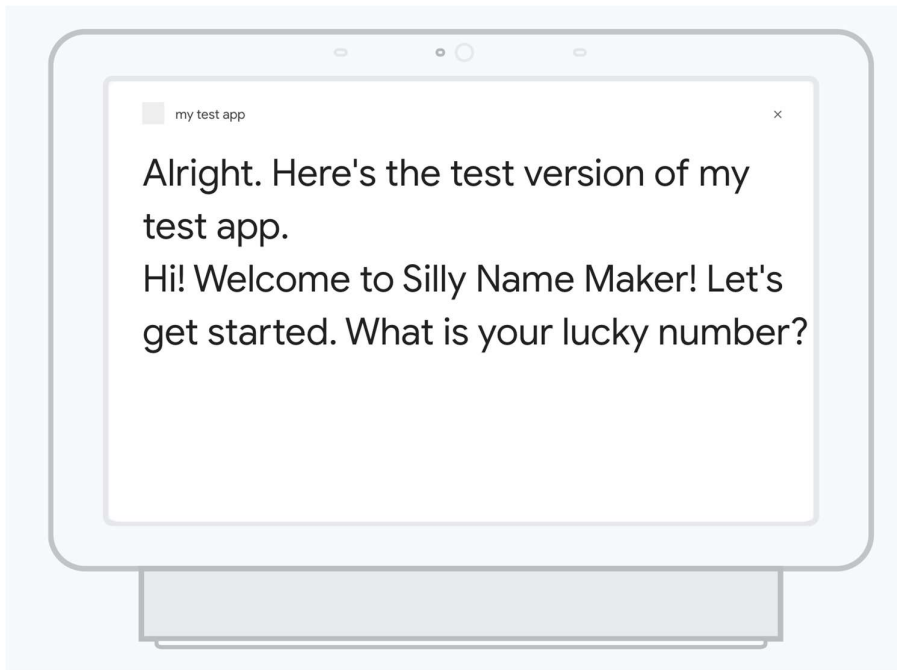
Specify intents that trigger "deep-link" actions in your app, allowing users to invoke specific functionality, such as "OK Google, ask Personal Chef for a hot soup recipe". Providing good action phrases. [Learn more.](#)

 **Auto-preview changes** 

Dialogflow will propagate changes to the Actions Console and Assistant Simulator automatically.

[CLOSE](#) [TEST](#) [MANAGE ASSISTANT APP](#) 

To invoke the Action, hit the enter key in the **Talk to my test app** box of the simulator console. You should be presented with a similar response:



Now enter in a number. When prompted to enter in a favorite color, enter in a color. You should be returned with a silly name built from the number and color you entered in:

[Reset Test](#) [View logs in Google Cloud Platform](#)

Talk to my test app

Alright. Here's the test version of my test app.
Hi! Welcome to Silly Name Maker! Let's get started. What is your lucky number?

12

What's your favorite color?

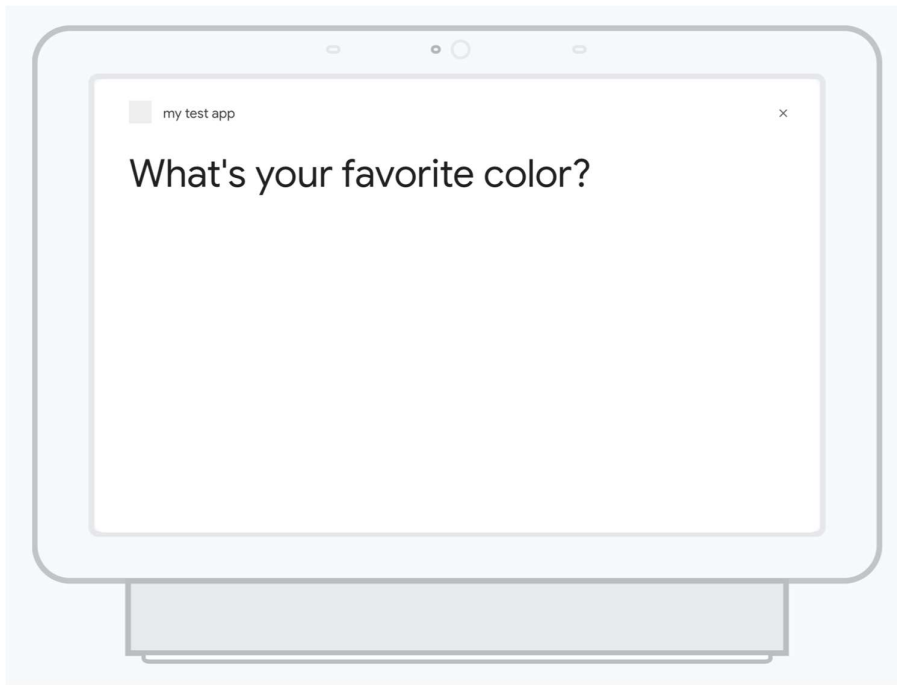
Red

Alright, your silly name is red 12! I hope you like it. See you next time.

'my test app' left the conversation

Now try entering in some input that follows the unexpected path. Enter `Talk to my test app` in the input area once more. Then provide a number when asked to enter in your lucky number.

When asked for your favorite color, try entering in a word that isn't a color. The reprompt phrase should kick in, asking you for your favorite color again:



Once you provide the correct input, your silly name will be generated and the conversation will end.

Optional: test your application on a Google Home device

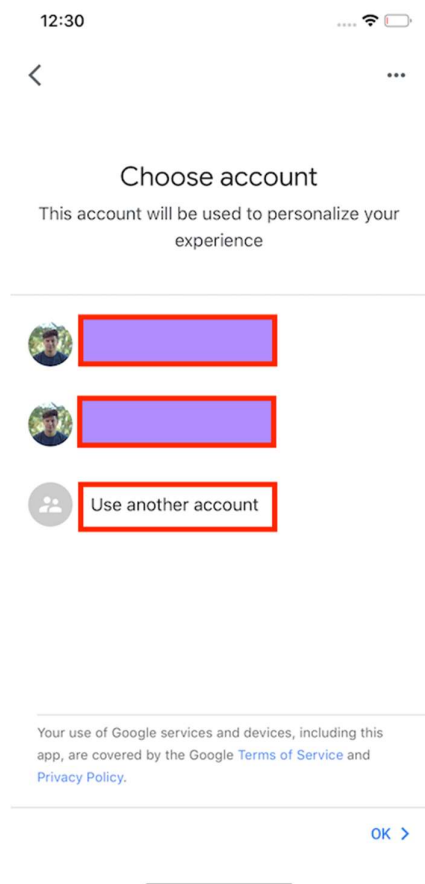
In this optional section, you will learn how to test an application on a Google Home device. At the risk of stating the obvious, **you must have a Google Home device to complete this step**. You will also need a smart phone (Android or iOS) to configure your Google Home. The following walkthrough uses an iPhone. If you use Android the steps may be slightly different.

Make sure that you have a new device or one that is factory reset before you begin this step. [This guide](#) will teach you how to reset all Google Home devices.

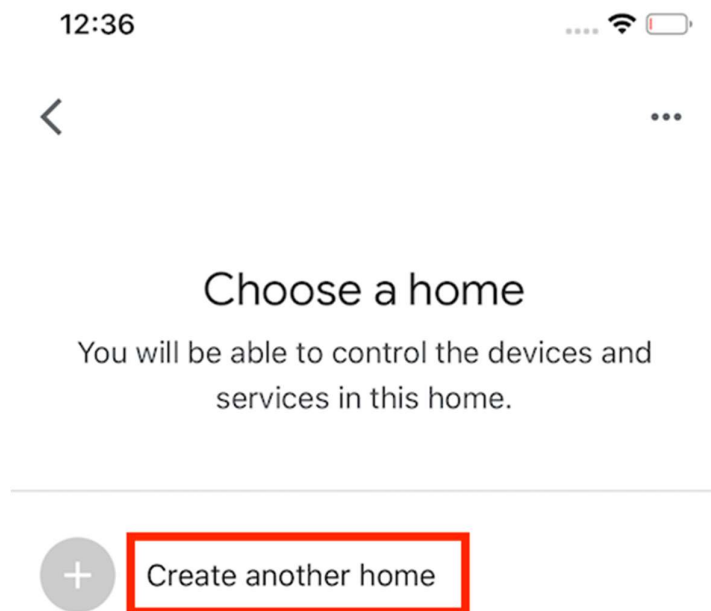
Configure your Google Home

Ensure that your device is plugged into an outlet and download the Google Home application [for android](#) or [iOS](#). Ensure that your smartphone's Wi-Fi and Bluetooth are turned on.

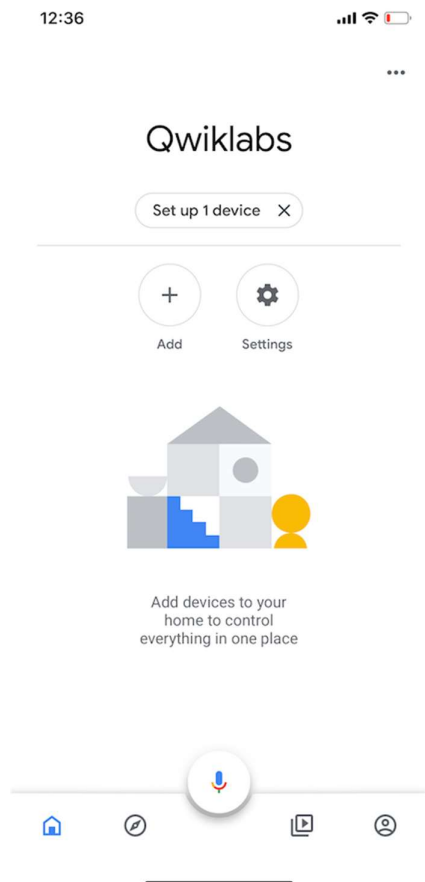
Once the application has downloaded, open it and click **GET STARTED**. Then for the **Choose account** section, select **Use another account > OK**:



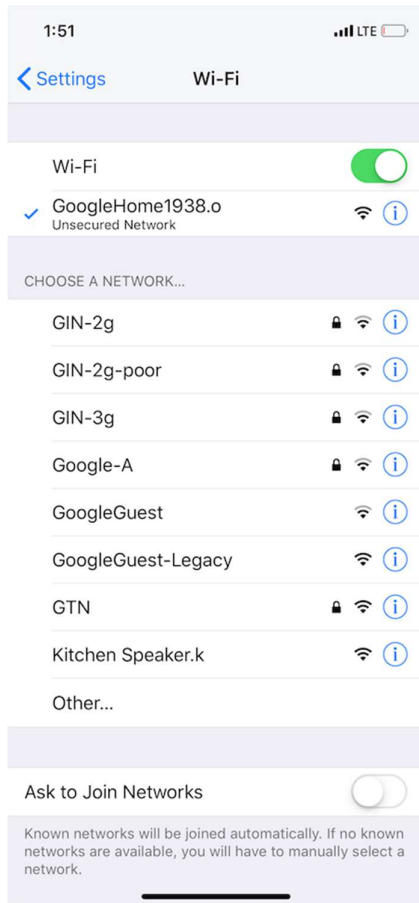
Then sign in using your Qwiklabs username and password. You will then be prompted to Choose a Google Home device. Select **Create another home** > **Next**:



Now give your Google Home a nickname (lab instructions use `Qwiklabs`.) Your page should now resemble the following:



Now, **open your Wi-Fi settings** on your smart phone. You should see a connection like GoogleHomeXXXX.x. Connect to this Wi-Fi network:



Note: the above screenshot comes from an iPhone. Android will differ.

Now return to the Google Home application. Your application should automatically detect the Google Home device and will attempt to connect to it. Once connected, you will be asked if you heard the test sound:

12:44



Did you hear the sound?

This lets you know you're connected to the
right Google Home Mini



TRY AGAIN

YES >

If so, select **YES**. Decide whether or not you want to help improve your Google Home by sending crash reports. When prompted, select where your device is located (e.g. the kitchen) and click **NEXT**.

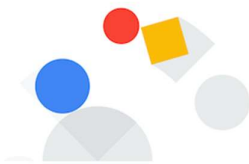
Now you will be asked to select a Wi-Fi network. Choose your local network. You will then see the following "Connecting to Wi-Fi" screen:

4:41



...

Connecting to Wi-Fi...



Once the device is connected, you will be asked to "Set up Google Assistant". Click **NEXT**.

Note: if this step fails and you get an "operation couldn't be completed" error, try clicking **NEXT** once more. If this doesn't work, you may have to run through the steps in this section once more.

Decide whether or not you want Google Assistant to recognize your voice. Decide whether you want to get personal results. Specify whether or not you want to use Voice 1 or Voice 2 for your Assistant. Enter your address.

Say **NOT NOW** when prompted to add media services. Say **NO THANKS** when prompted to receive email updates. When you land on the "Almost done!" page, click **NEXT**. You should now receive a prompt that says your device is ready:

12:50



Kitchen Speaker is ready

Let's get started



CONTINUE >

Click **CONTINUE**. Follow any extra set up prompts.

Now that you have set up your Google Home with your Qwiklabs account, you're all ready to test your application.

Test the application on your Google Home

Now that your device is configured with your Qwiklabs account, you can test your application using voice commands.

Say the following to your Google Home:

"Okay Google, talk to my test app".

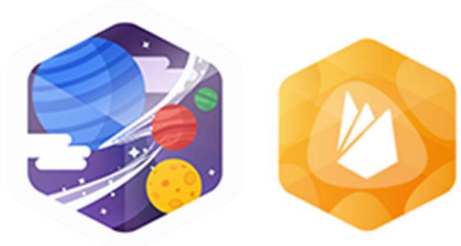
You will then hear "Sure, here's the version of of my test app." This will be followed by the welcome intent: "Hi, welcome to Silly Name Maker! Let's get started. What is your favorite number?"

You can now follow the prompts to generate a silly name.

Congratulations!

In this lab, you built a robust Google Assistant application with Dialogflow and Cloud Functions. After creating an Actions project, you configured and built two intents. From there you learned how to initialize a Cloud Function, where you added your own fulfillment logic and packages. After deploying the function, you added a webhook and tested your Assistant application with the Actions simulator. You are now ready to take more advanced Google Assistant labs.

Finish Your Quest



This self-paced lab is part of the Qwiklabs [OK Google: Build Interactive Apps with Google Assistant](#) and [Build Apps & Websites with Firebase](#) Quests. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. [Enroll in a Quest and get immediate completion credit if you've taken this lab. [See other available Qwiklabs Quests](#).

Take Your Next Lab

Continue your Quest with [Google Assistant: Build a Youtube Entertainment App](#), or check out these other suggestions:

- [Google Assistant: Build a Restaurant locator with the Places API](#)
- [Build a Serverless Web App with Firebase](#)

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 November 7, 2020

Lab Last Tested November 7, 2020

Copyright 2021 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.