

## Google Home Project Handout:

Randyll Bearer, Raj Patel, Zachary Barlotta

### Technologies Used:

#### - Google Chrome Browser:

- Location Enabled
- Version 62.0.3202.94

#### - Actions:

- Google Assistant Proprietary Dev./Testing Software
- console.actions.google.com

#### - Dialogflow:

- Google Assistant Proprietary Dev. Software
- console.dialogflow.google.com

#### - Nodejs:

- JavaScript Framework
- Version 6.11.4

#### - Firebase:

- Cloud Function Deployment Software
- Version 3.13.1

#### - Github:

- Version Control Software
- Our Repository: <https://github.com/rlb97/Capstone-Project>

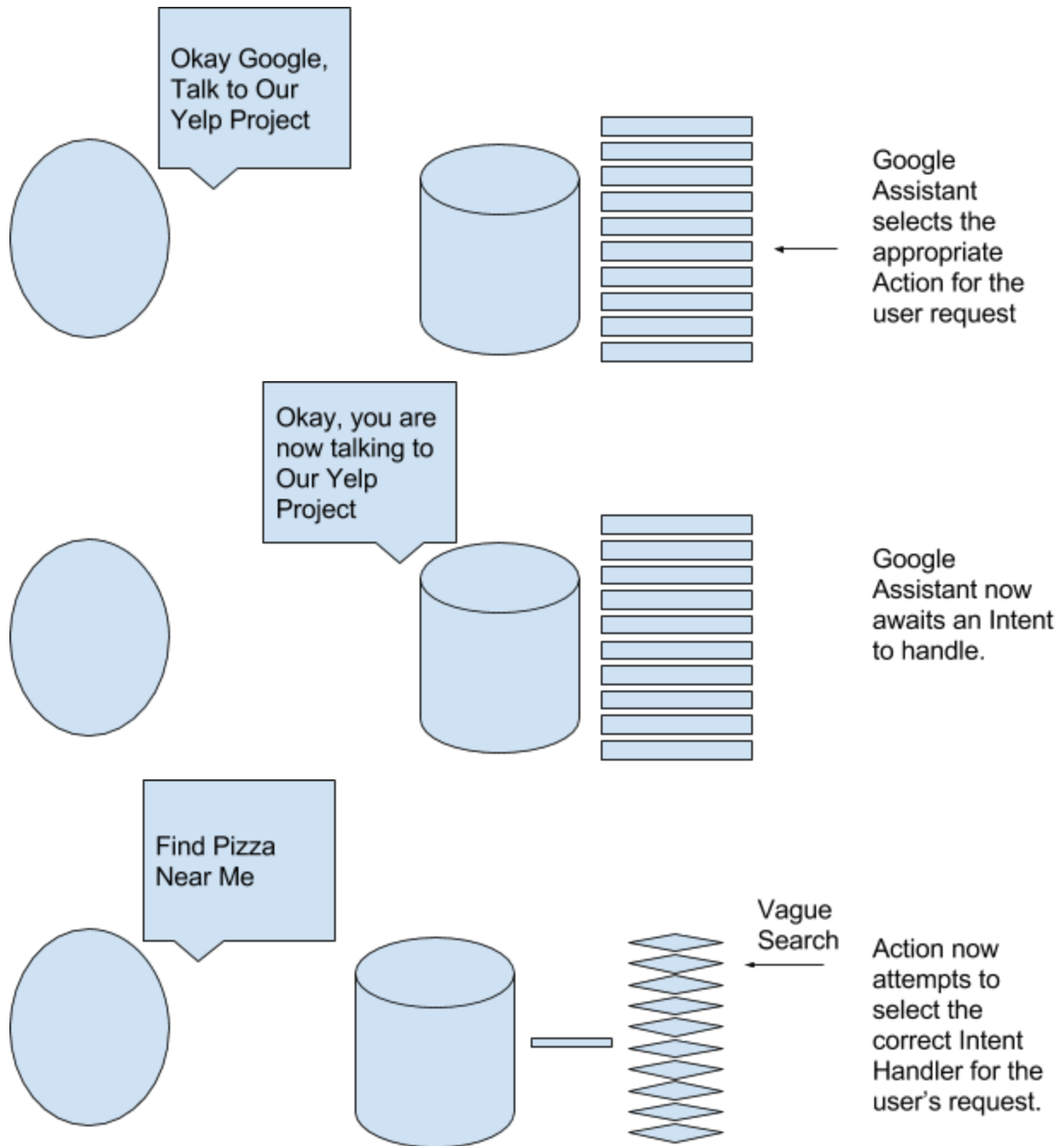
### Helpful Links:

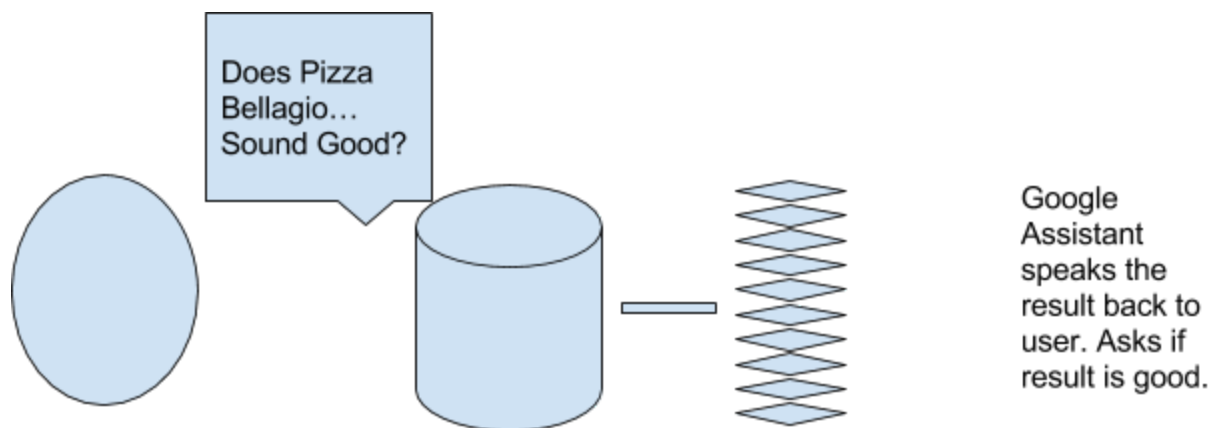
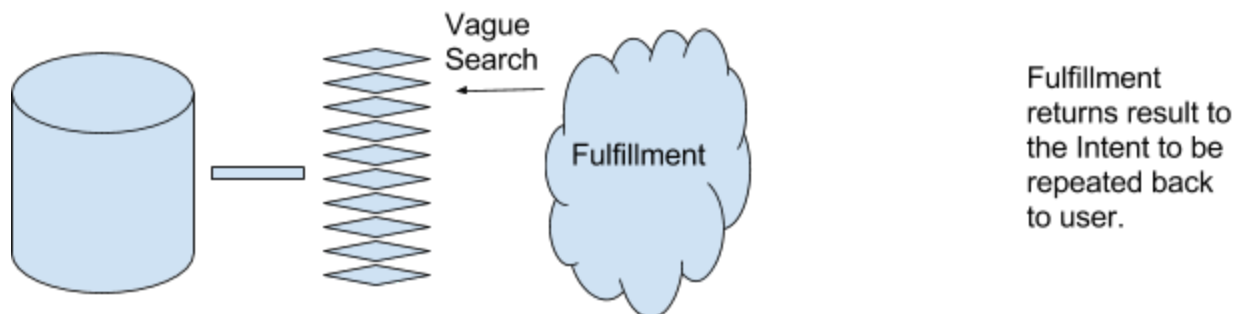
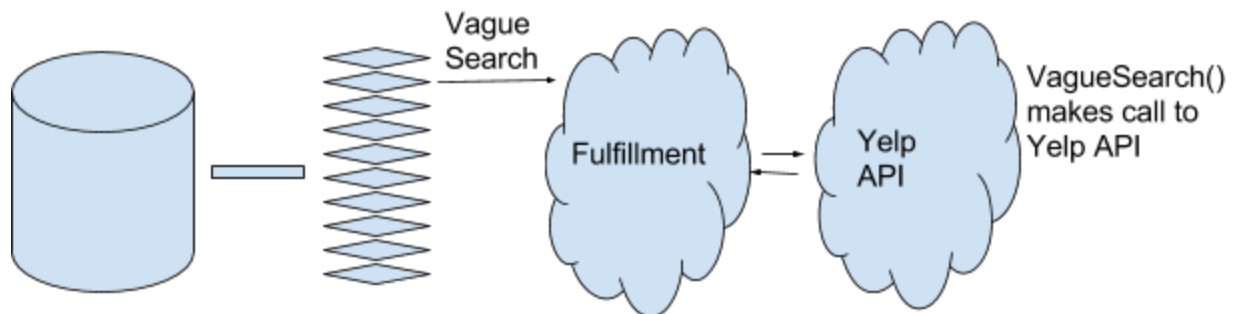
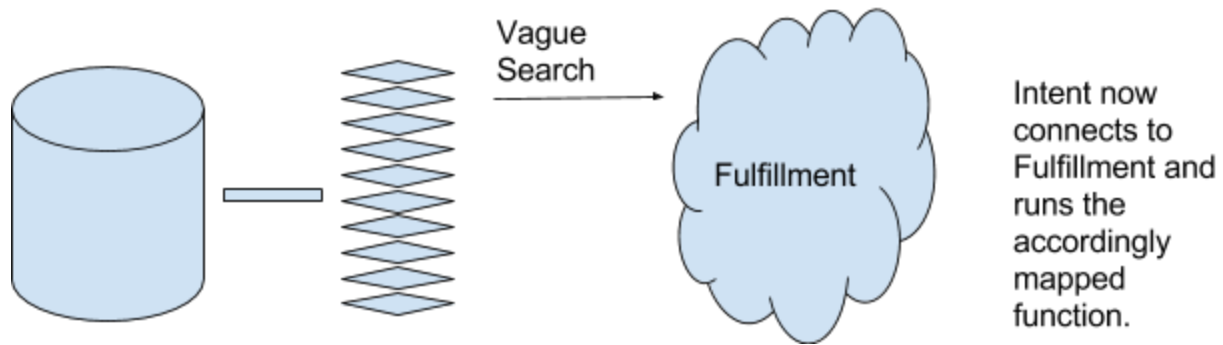
- <https://developers.google.com/actions/dialogflow/first-app>
- <https://miningbusinessdata.com/step-by-step-guide-to-api-ai/>
  - Refers to Dialogflow as API.ai (the old service before it was replaced) but has Updated examples for Dialogflow.

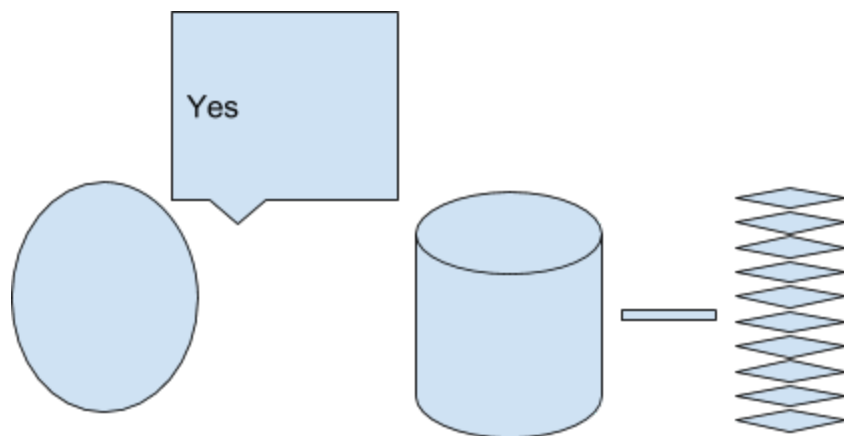
## Terminology Used:

- **Google Assistant:** Google's Voice Assistant AI. Can be run on top of any semi-modern Android device. Used in this project as a way to facilitate communication with a user through voice-to-text and text-to-voice conversions. Provides both text and voice feedback. Can be tested and simulated through the `console.actions.google.com` emulator. Communicates with the **Fulfillment** through JSON interchange.
- **Google Home:** Google's hands-free and text-free home device, runs the **Google Assistant** to handle all functionality. Our goal for this project was to allow our software to be ran on this device.
- **Action:** A specific 'skill' that the **Google Assistant** possesses which can handle a specific user request, i.e. 'Check the Weather', 'Play a Song', or 'Talk to Our Yelp Project'. Upon receiving a user request the **Google Assistant** tries to match that request with the best matching **Action**, which then asks the user for an **Intent**. Can be created through `console.actions.google.com`.
- **Intent:** A conversation handler which handles various user requests for a given **Action**, i.e. 'Find Pizza near me' or 'Look up Piada'. An **Action** can have any number of **Intents**, allowing an **Action** to handle any number of requests. Once selected, an **Intent** will fulfill a user request by calling the appropriately mapped function in the **Fulfillment**. Can be created through the **Dialogflow** software at `console.dialogflow.google.com`.
- **Fulfillment:** The remotely hosted code which handles user requests. Composed of cloud functions which have been mapped to **Dialogflow** Intents. Handles all 'Program Logic' and does the majority of the work for the software. For our project, our **Fulfillment** was programmed in **Nodejs** and deployed through **Firebase**.
- **VagueSearch():** One of the three implemented searches in our **Action**. **VagueSearch()** allows for the user to search the Yelp API for vague categories of food, i.e. Italian, Sushi, Steakhouse.
- **DirectSearch():** One of the three implemented searches in our **Action**. **DirectSearch()** allows for the user to search the Yelp API for specific businesses, i.e. Piada, McDonald's, Union Grill.
- **RandomSearch():** One of the three implemented searches in our **Action**. **RandomSearch()** returns a randomly selected restaurant near the user based off of the Yelp API's `best_match` feature.
- **ActionMap:** A Map object created in the **Fulfillment**. Maps **Action Intents** to functions in the **Fulfillment** so that the appropriate code gets called for each **Intent**.
- **Entity:** A variably-filled keyword which triggers **Intents**.

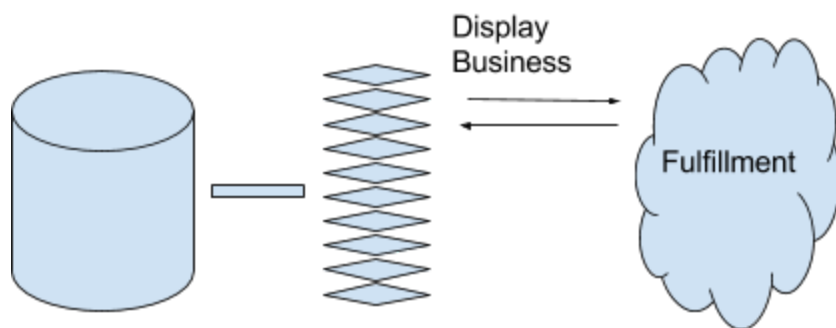
## Sample Conversation



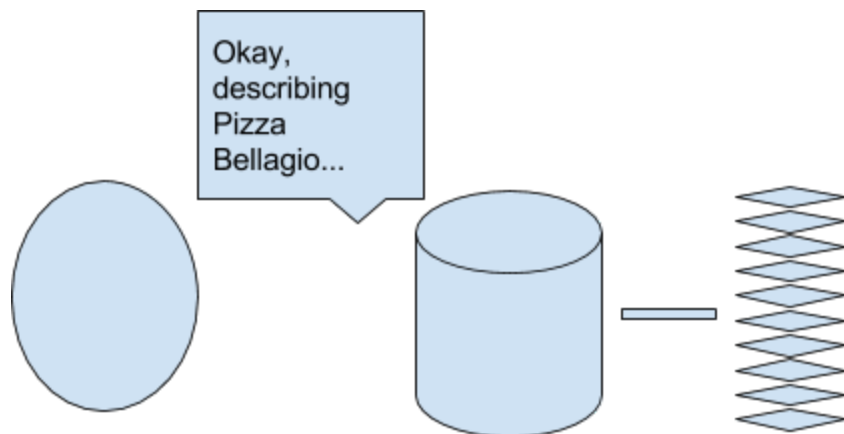




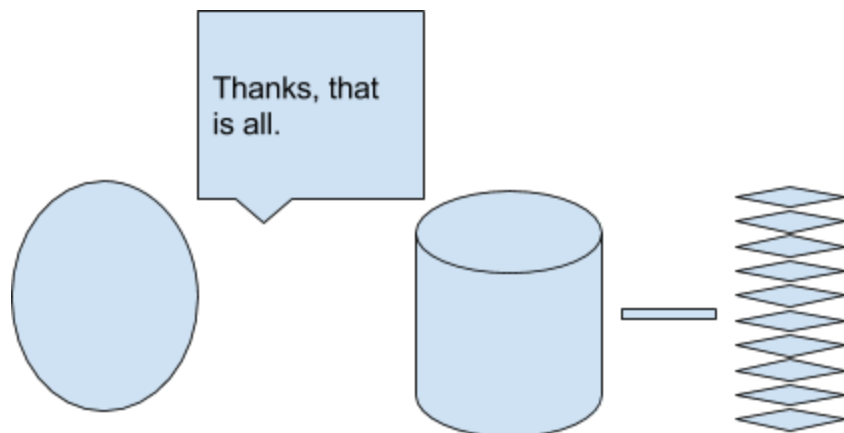
User responds to the result, Intent now prepares to print all selectedBusiness details



Action selects the DisplayBusiness Intent and calls the Fulfillment function to return more specific details.



Google Assistant repeats the details provided by the Fulfillment



Our Yelp Action shuts down.

## Dialogflow Rundown: Main

1. A text parser for **Intent** testing purposes. Will show you which **Intent** will handle your input string.

2. **Dialogflow** sidebar, allows you to switch between different tabs. This screenshot was taken on the **Intents** tab.

3. List of **Intents** set up for this specific **Action**.

## Dialogflow Rundown: Intent

The screenshot shows the Dialogflow console interface for configuring an intent. The left sidebar contains navigation options: Intents (selected), Entities, Training [beta], Integrations, Analytics [new], Fulfillment, Prebuilt Agents, and Small Talk. The main area is titled 'vague\_search 1.' and includes a 'SAVE' button. It displays a list of user expressions that trigger the intent, such as 'Discover Steakhouse', 'Look up Sushi Bars', 'Is there Italian around me', 'I want pizza', 'Find me Chicken Wings', 'Find Chinese food near me', and 'Is there Mexican near here'. Below this is a table of entities extracted from the expressions:

PARAMETER NAME	ENTITY	RESOLVED VALUE
restaurants	@restaurants	Mexican

The 'Events' section is empty. The 'Action' section shows the function 'vague\_search' mapped to the fulfillment. Below this is a table of parameters for the fulfillment function:

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST	PROMPTS
<input checked="" type="checkbox"/>	restaurants	@restaurants	\$restaurants	<input type="checkbox"/>	What kind of f o...
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>	—

The 'Fulfillment' section shows the option 'Use webhook' selected, with a note '6.' next to it. The 'Response' section is empty.

1. Name of the **Intent** you are working on.
2. List of user requests which will trigger the **Intent**.
3. List of **Entities** which the user request must include.
4. Name of the function to be mapped in the **Fulfillment**
5. List of arguments to be passed to the **Fulfillment**. We pass the “Restaurants” **Entity** here so that we know which specific restaurant the user wants to look for in the **Fulfillment**.
6. “Use webhook” Tells our **Intent** to run the external code in the **Fulfillment**.

## Dialogflow Rundown: Entities

Dialogflow

restaurants

SAVE

RandyTest (owned b...)

en

Intents

Entities

Training (beta)

Integrations

Analytics (new)

Fulfillment

Prebuilt Agents

Small Talk

Docs

Forum

Support

Account

Logout

☒ Define synonyms ☒ Allow automated expansion

1.

Search entries

5 OF 5


Fish	Fish, Seafood
Gelato	Gelato
Juice	Juice, Juice Bar, Lemonade
Milkshake	Milkshake, Ice Cream, Dessert
Pretzels	Pretzels
Shaved Ice	Shaved Ice, Shaved Snow, Italian Ice, Snow Cones
Smokehouse	Smokehouse, Roadhouse, Steakhouse, smoke house, road house, steak house
Candy store	Candy
Cheese Store	Cheese Store, Cheese
Chocolate	Chocolate
Fruit	Fruit, Produce, Vegetable
Pasta	Pasta
Popcorn	Popcorn
Tofu	Tofu
Sugar Shacks	Sugar Shacks, Sugar Shack
Wineries	Wineries, Winery
Fast Food	Fast Food
Sea Food	Sea Food
Soul Food	Soul Food
Steak	Steak, Steak House


Click here to edit entry


1. “Allow automated expansion” tells **Dialogflow** to go and provide their own synonyms.
2. The specific word which will fulfill the **Entity** requirement for the user request to be handled by an **Intent**. For example, if the **Intent** is looking for a request such as “Find @restaurant near me”, then the user request “Find Milkshake near me” would trigger the **Intent**.
3. The list of synonyms which will also trigger the **Intent** but pass the value of the parameter name. I.e., “Shaved Snow” would pass “Shaved Ice” to the **Fulfillment**.







## Dialogflow Rundown: Fulfillment


 Dialogflow


RandyTest (owned b... 


en 


 Intents 


 Entities 


 Training <sup>[beta]</sup>

 Integrations

 Analytics <sup>[new]</sup>


 **Fulfillment**


 Prebuilt Agents


 Small Talk


> Docs


> Forum

 Support

 Account

 Logout

 **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](#).

[Webhook example](#)


URL\*  **1.**

BASIC AUTH


HEADERS

 Add header

DOMAINS

Inline Editor (Powered by Cloud Functions for Firebase) DISABLED 

Build and manage fulfillment directly in Dialogflow via Cloud Functions for Firebase. [Docs](#)

index.js package.json 

```
1 'use strict';
2
3 const functions = require('firebase-functions'); // Cloud Functions for Firebase library
4 const DialogflowApp = require('actions-on-google').DialogflowApp; // Google Assistant helper lib
5 exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request, response) => {
6   console.log('Dialogflow Request Headers: ' + JSON.stringify(request.headers));
7   console.log('Dialogflow Request body: ' + JSON.stringify(request.body));
8   if (request.body.result) {
9     processV1Request(request, response);
10  } else if (request.body.queryResult) {
11    processV2Request(request, response);
12  } else {
13    console.log('Invalid Request');
14    return response.status(400).end('Invalid Webhook Request (expecting v1 or v2 webhook request)');
15  }
16 });
17 /*
18  * Function to handle v1 webhook requests from Dialogflow
19  */
20 function processV1Request (request, response) {
21   let action = request.body.result.action; // https://dialogflow.com/docs/actions-and-parameters
```

1. The URL which points to your cloud hosted **Fulfillment**. For our project we got this URL from **Firebase** automatically after deployment.

## Dialogflow Rundown: Integrations

Dialogflow

Integrations

RandyTest (owned b...

en

Intents

Entities

Training [beta]

**Integrations**

Analytics [new]

Fulfillment

Prebuilt Agents

Small Talk

> Docs

> Forum

Support

Account

Logout

### One-click integrations

1. Google Assistant SETTINGS	2. Web Demo SETTINGS	 Facebook Messenger 	 Slack 
 Viber 	 Twitter 	 Twilio IP 	 Twilio (Text messaging) 
 Skype 	 Tropo (Text messaging) 	 Telegram 	 Kik 
 LINE 	 Cisco Spark 	 Amazon Alexa 	 Microsoft Cortana 

### Dialogflow SDK's

--	--	--	--

1. To update the version of your current **Action** that you want to test, you must click on “Google Assistant”, “Test”, then “View” to open a new tab with the simulator on it.

2. “Web Demo” must be activated in order to the simulator to work properly.

## Actions Simulator Rundown: Main

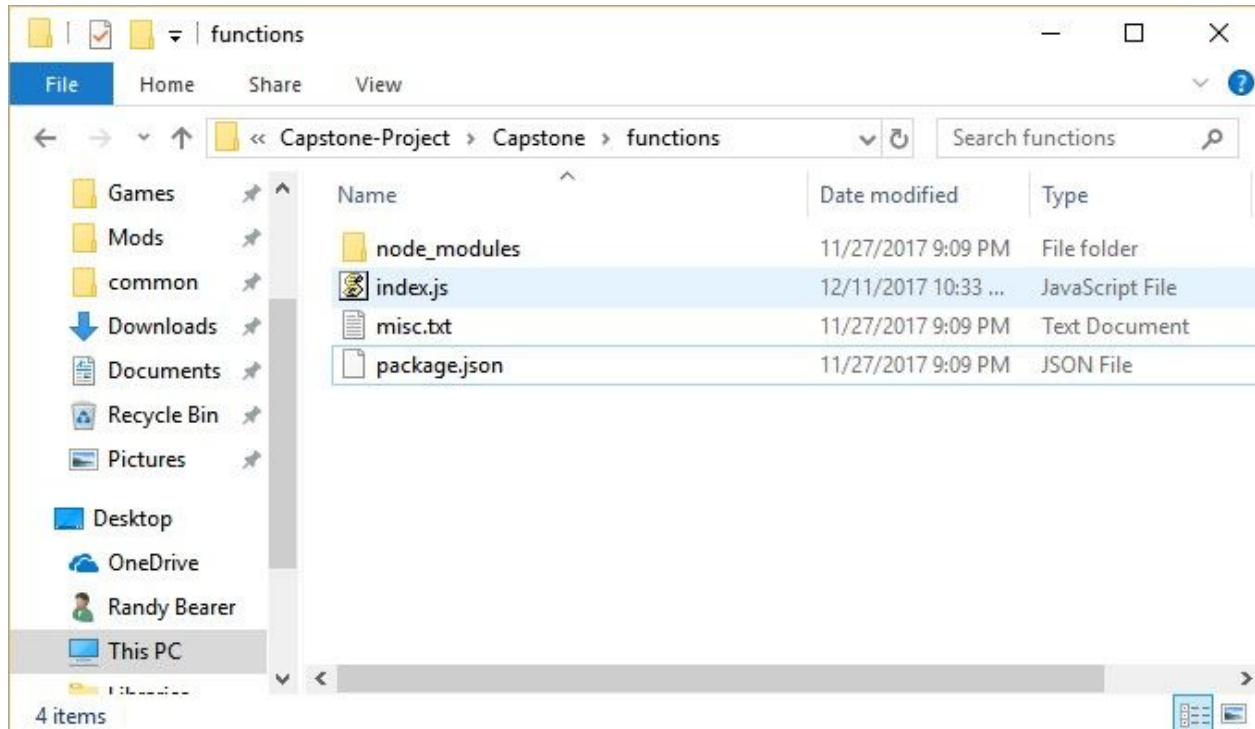
The screenshot displays the Actions on Google Simulator interface. On the left, a sidebar (labeled '3.') contains 'Analytics', 'Simulator', and 'Backend services'. The main area shows a conversation for 'Our Yelp Project' with a user input 'Talk to Our Yelp Project' and a simulated response. The right-hand window (labeled '2.') shows a 'REQUEST' tab with a JSON object containing metadata like 'conversationToken', 'debugLevel', 'inputType', 'locale', 'mockLocation', 'formattedAddress', 'query', and 'surface'. The top bar includes 'Actions on Google', 'Capstone', and 'Go to docs'.

1. Input console. Allows you to enter in requests through either text or speech.
2. Debug window. Each tab provides different debug information. Working as of 11/20/2017.
3. Sidebar. "Analytics" Allows you to edit the meta-information for your **Action**/Project, such as Title, author, description, summary, and also allows you to publish **Actions**.

## Firestore Rundown:

- To initialize **Firestore** for a directory, please refer to the first link under **Helpful Links** For a tutorial on getting the deployment service set up.
- To deploy the contents of the current working directory, use the following command:  
firebase deploy --only functions  
Other keywords after “--only “ may be used depending on your requirements, but for Our project we exclusively used “functions”.
  - **Firestore** will then display the results of your deployment attempt. If your Deployment was successful, then you will be displayed a URL to be copied Into the **Fulfillment** tab in the **Dialogflow** sidebar.
- To debug console output from the **Firestore** server, use the following command:  
firebase functions:log  
Which will print the results of your last run **Fulfillment**.

## Misc. Details:



- The folder/working directory you should try to deploy from should be named “functions” if you are attempting to “deploy --only functions”.
- Index.js will contain the code for you fulfillment. Other files may be used as long as they are included by Index.js.
- Package.json will contain the meta-information for your deployment. E.g. Author, Name, Dependencies, etc.
- Misc.txt should be used for any miscellaneous information you want to persist in the directory.