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.

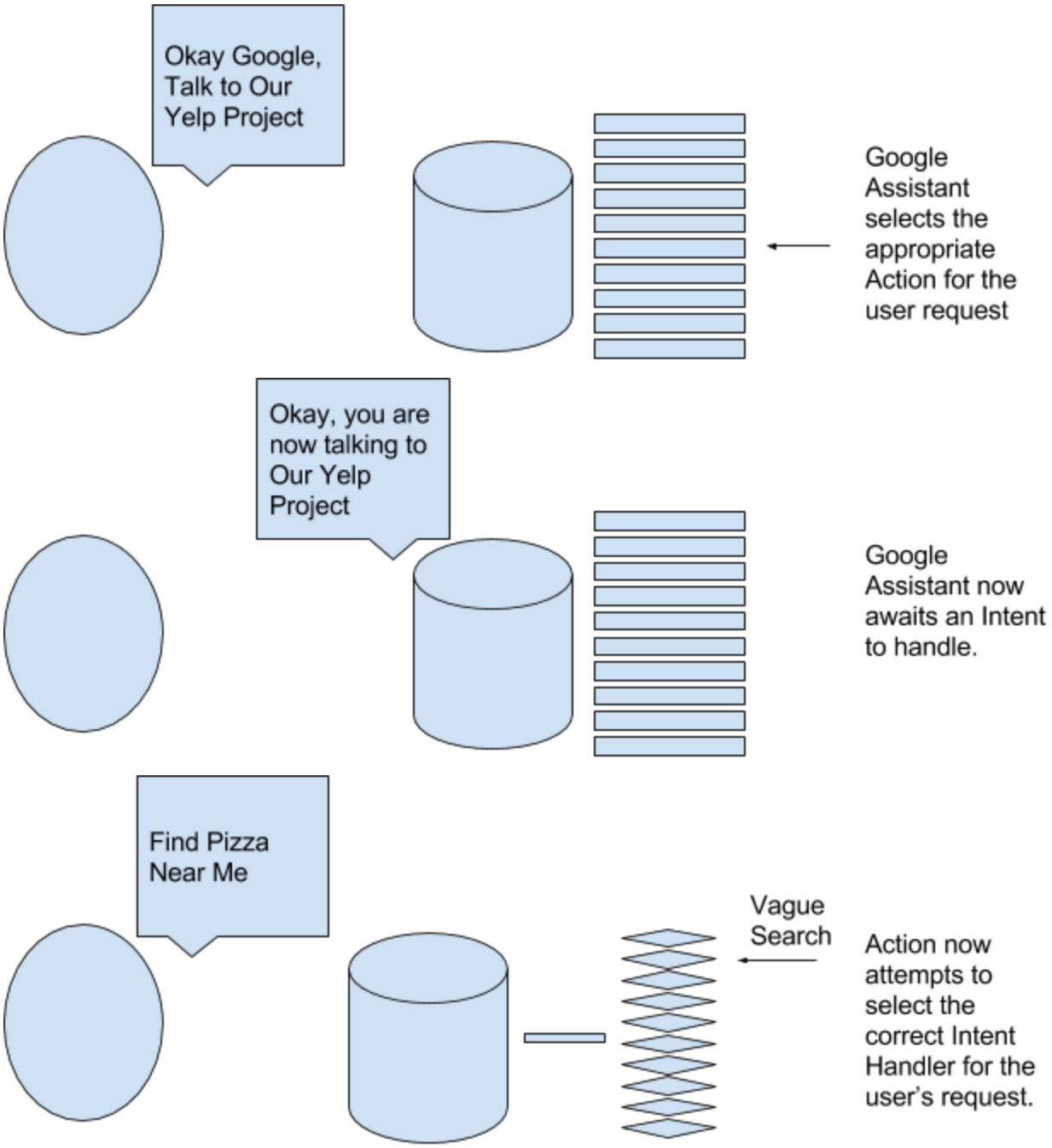
- **DirectSearch()**: One of the three implemented searches in our **Action**. **DirectSearch()** allows for the user to search the Yelp API for specific businesses, i.e. Piada, McDonald's, Union Grill.

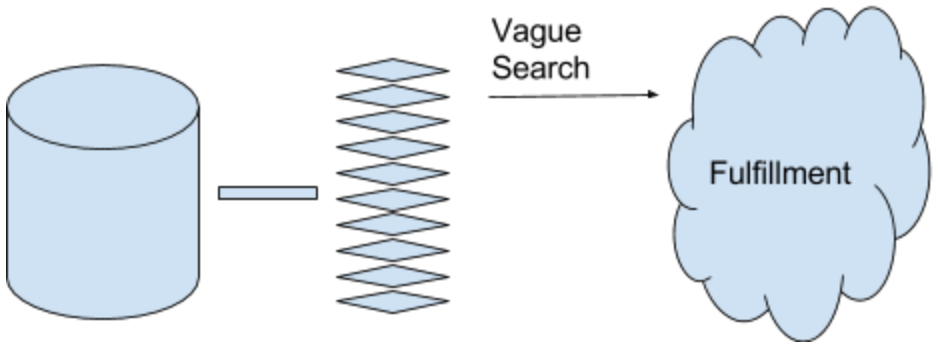
- **RandomSearch()**: One of the three implemented searches in our **Action**. **RandomSearch()** returns a randomly selected restaurant near the user based off of the Yelp API's best_match feature.

- **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**.

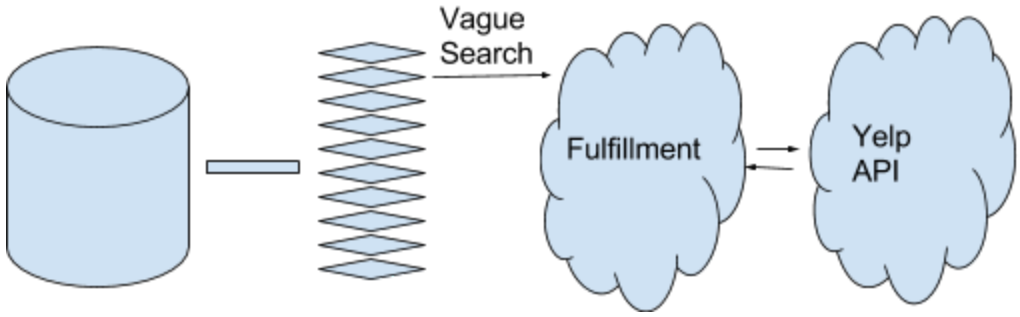
- **Entity:** A variably-filled keyword which triggers **Intents**.

Sample Conversation

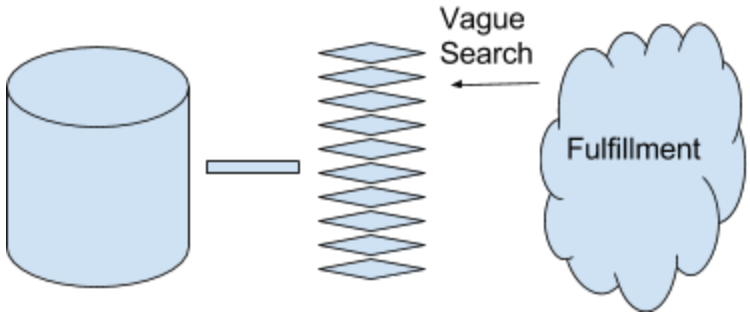




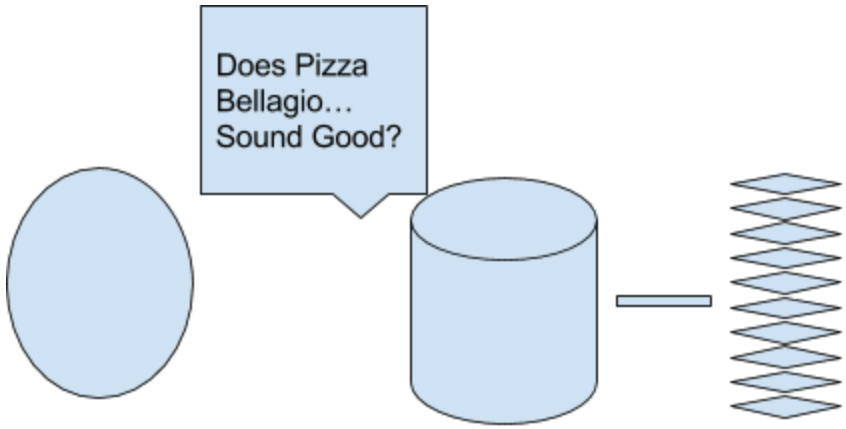
Intent now connects to Fulfillment and runs the accordingly mapped function.



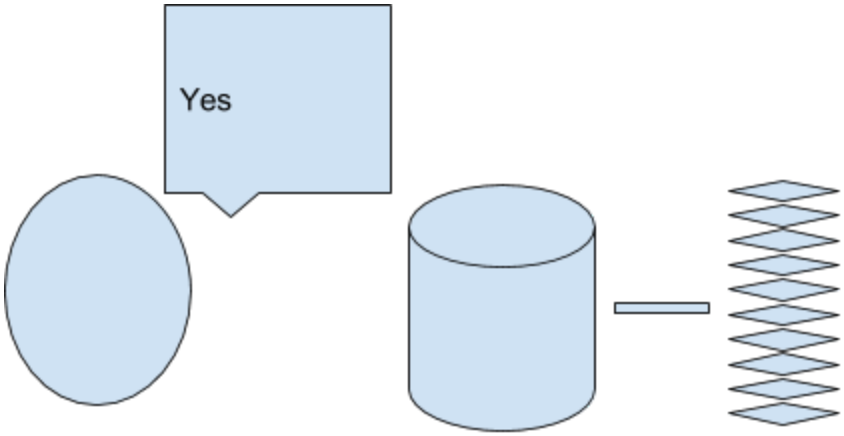
VagueSearch() makes call to Yelp API



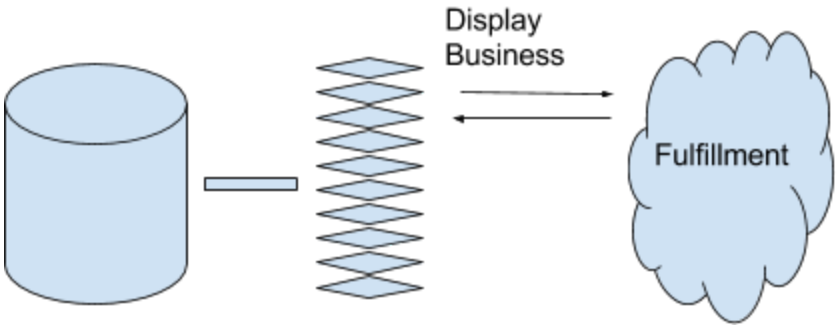
Fulfillment returns result to the Intent to be repeated back to user.



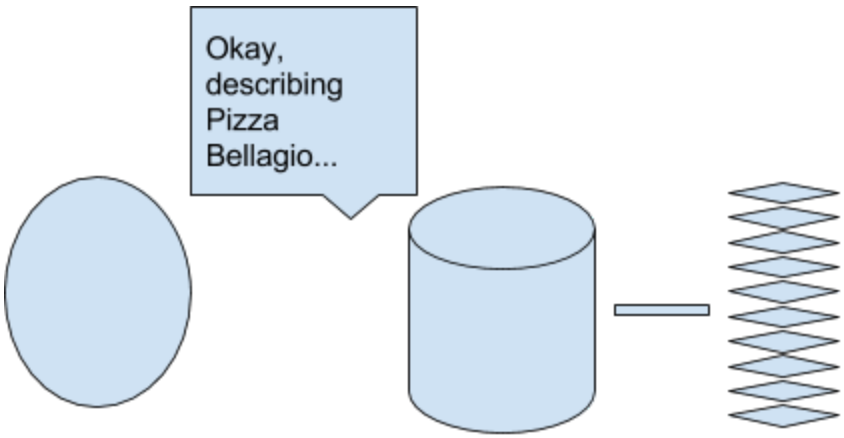
Google Assistant speaks the result back to user. Asks if result is good.



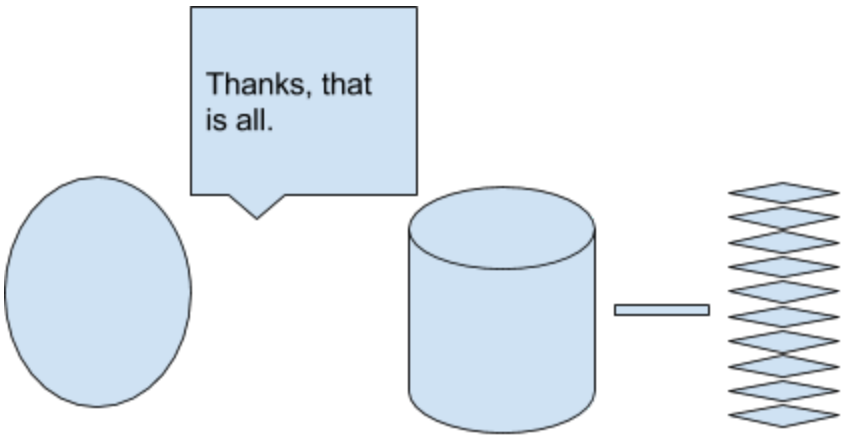
User responds to the result, Intent now prepares to print all selectedBusiness details



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

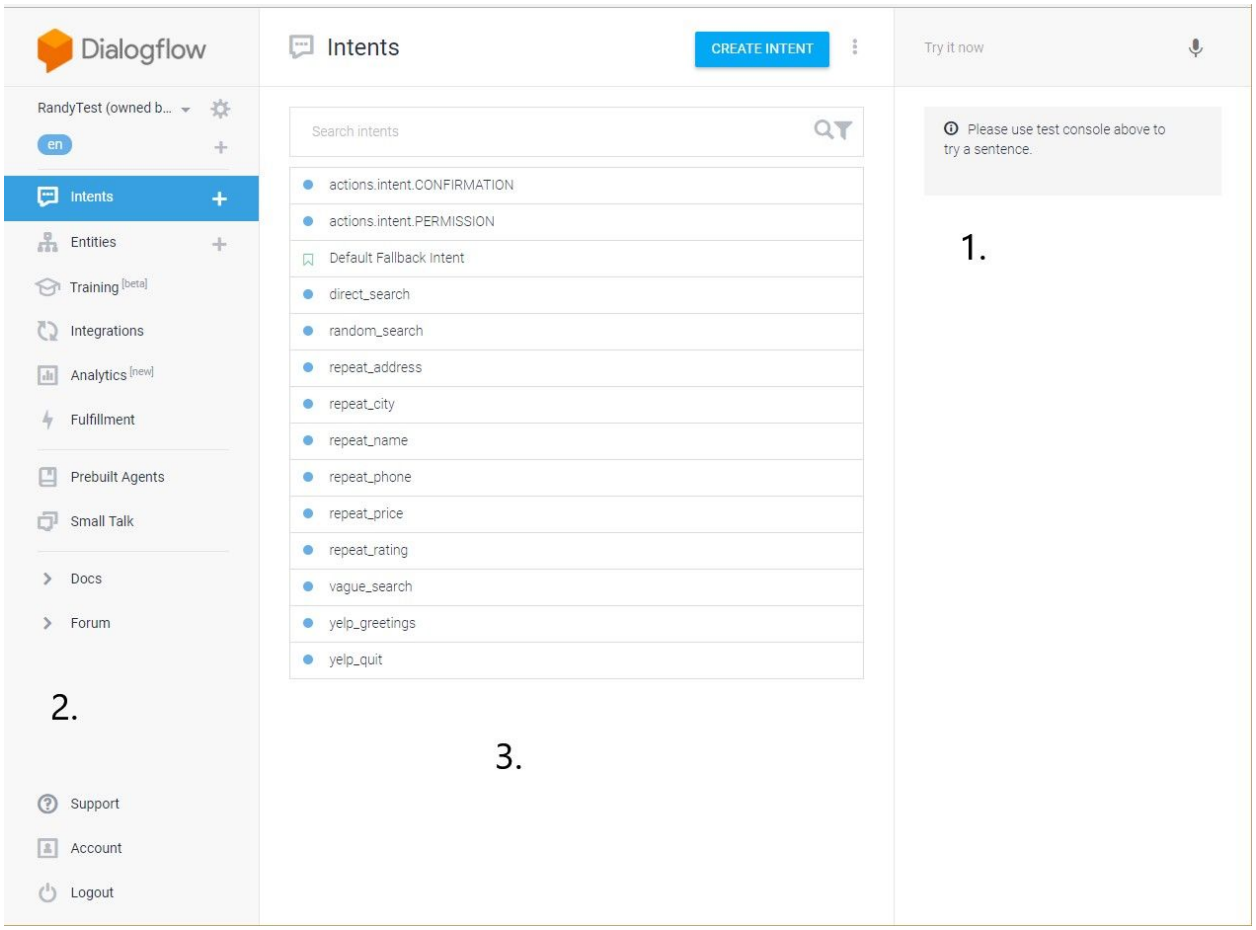


Google Assistant repeats the details provided by the Fulfillment



Our Yelp Action shuts down.

Dialogflow Rundown: Main

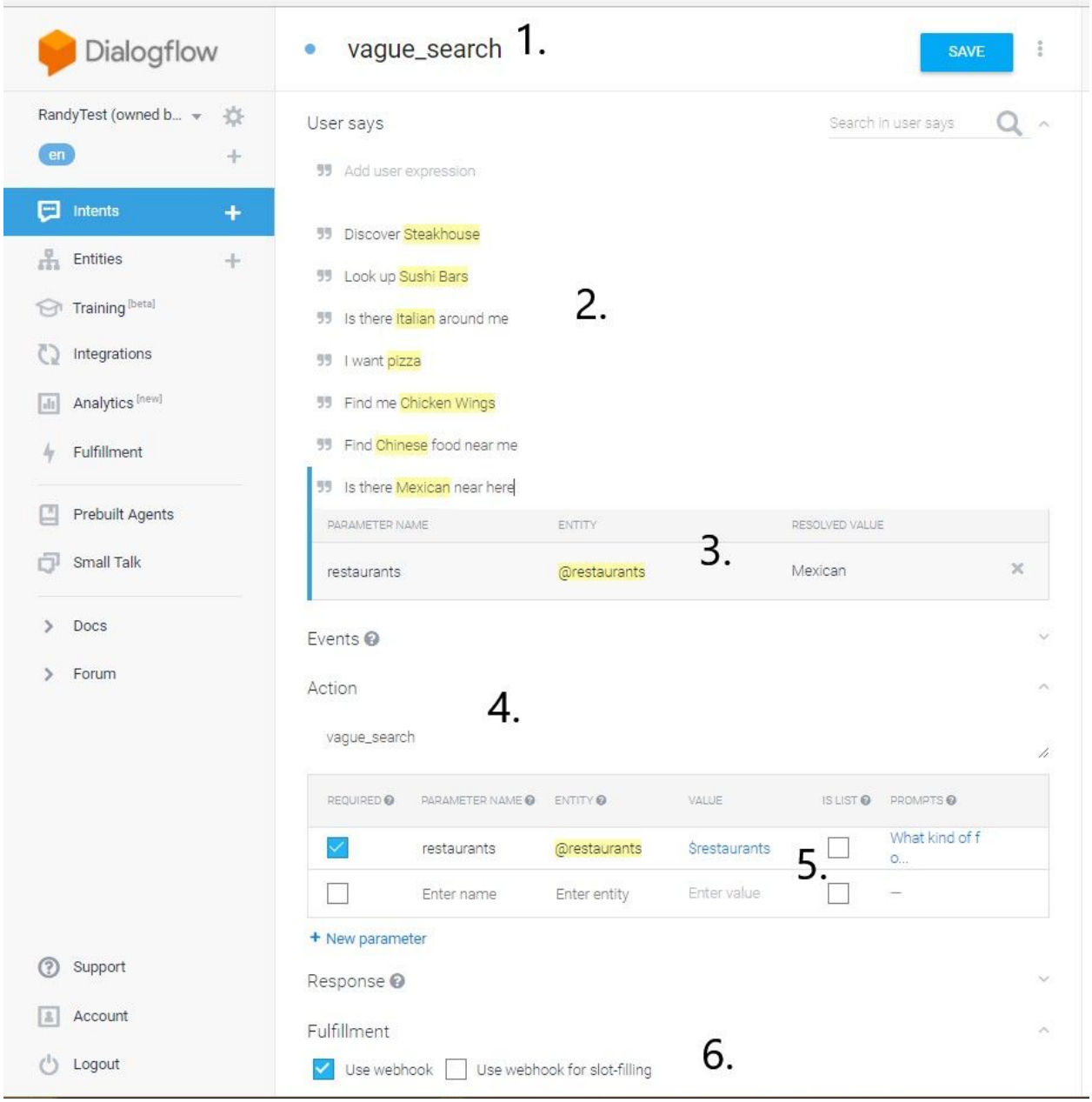


1. A text parser for **Intent** testing purposes. Will show you which **Intent** will handle your input string.

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

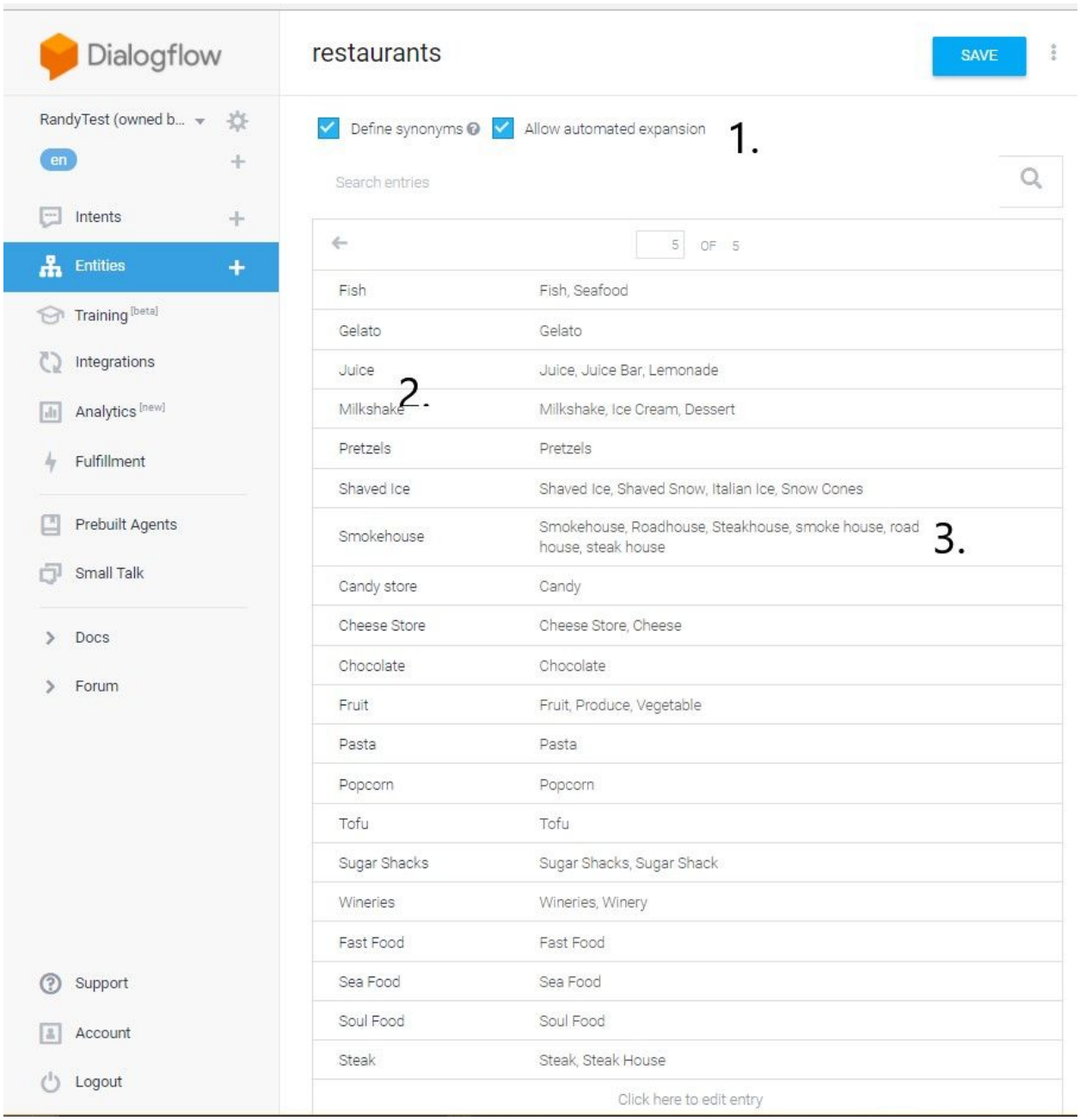
3. List of **Intents** set up for this specific **Action**.

Dialogflow Rundown: Intent



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

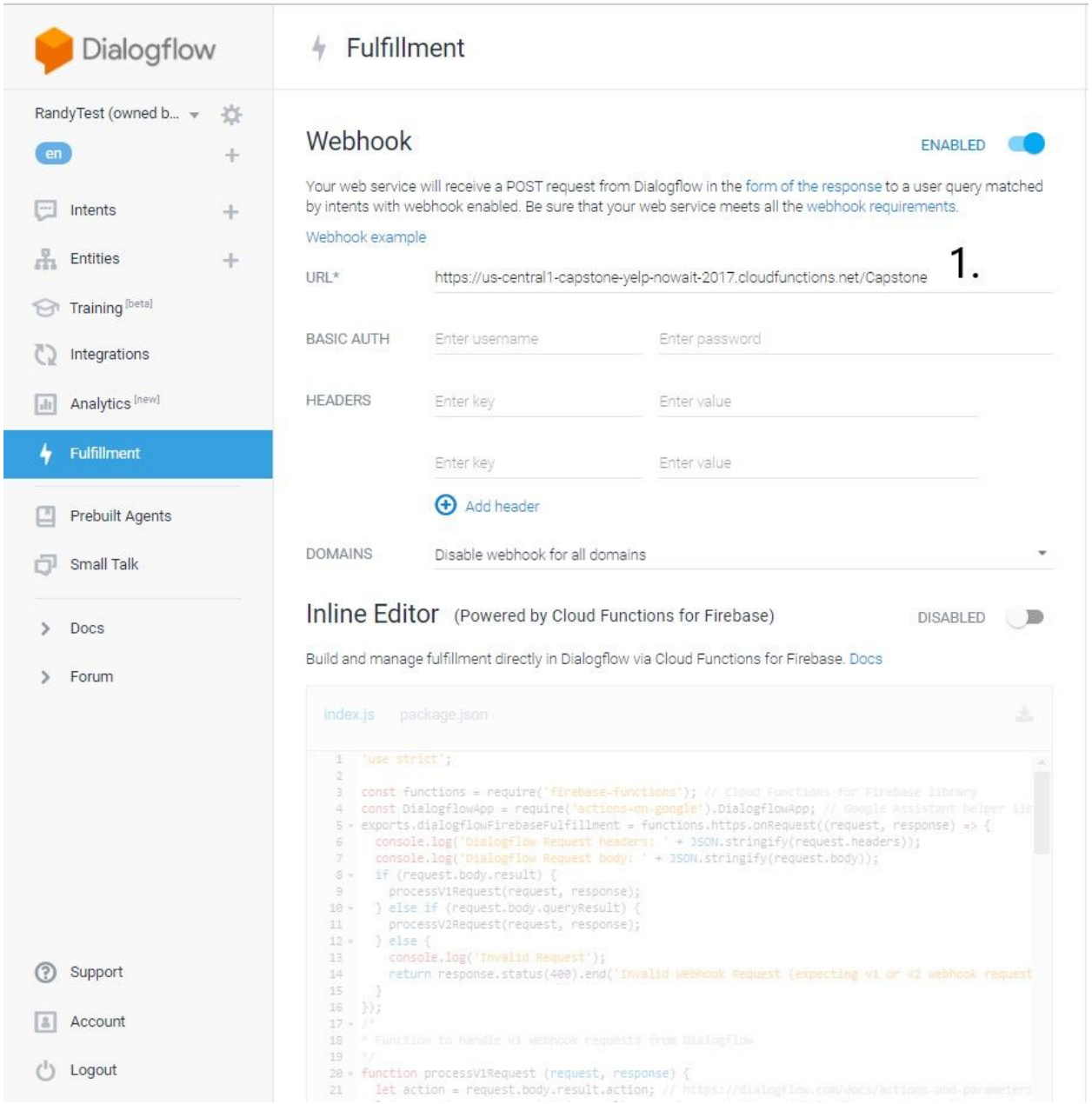


1. “Allow automated expansion” tells **Dialogflow** to go and provide their own synonyms.

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**.

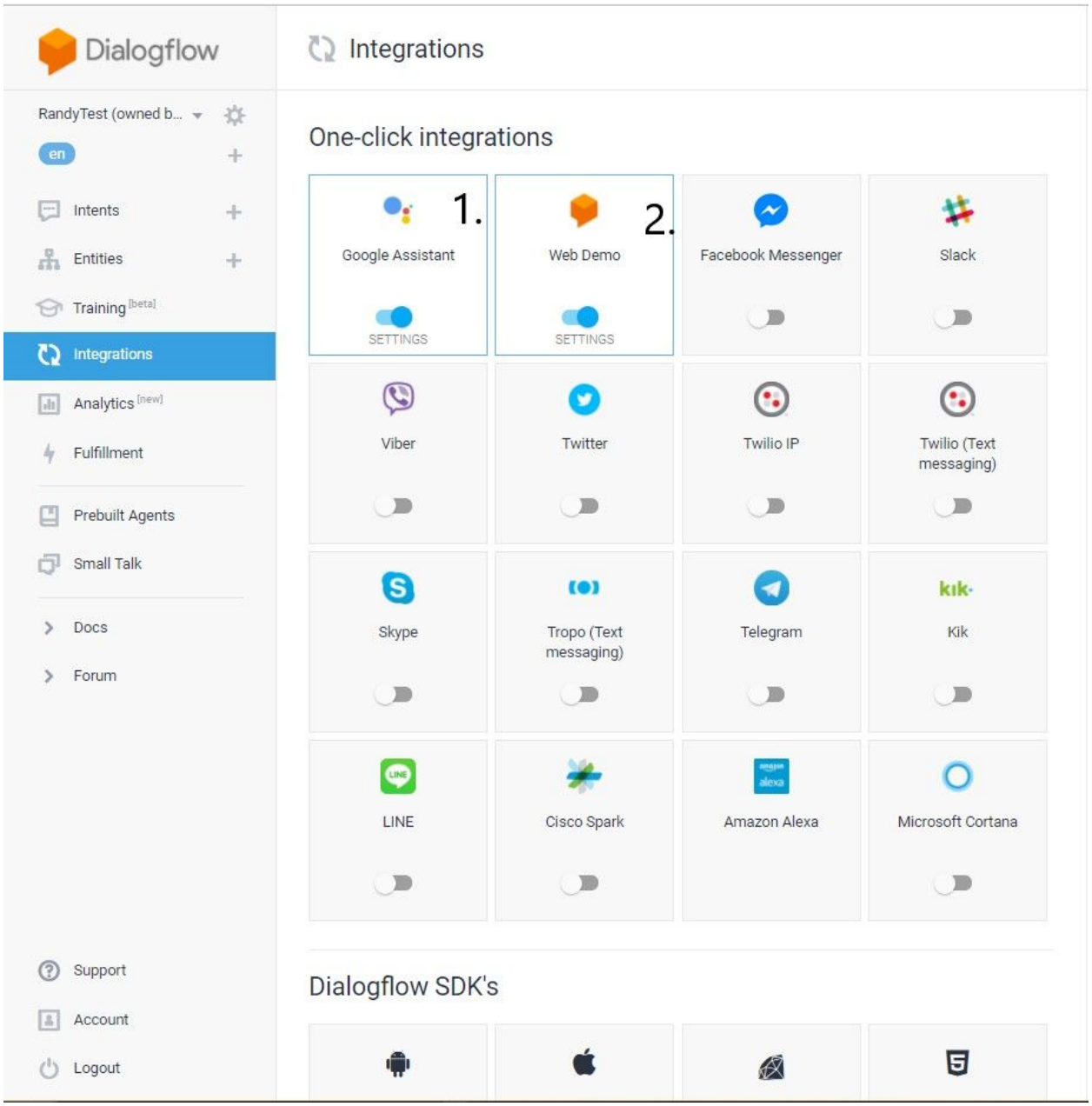
3. The list of synonyms which will also trigger the **Intent** but pass the value of the parameter name. I.e., “Shaved Snow” would pass “Shaved Ice” to the **Fulfillment**.

Dialogflow Rundown: Fulfillment



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

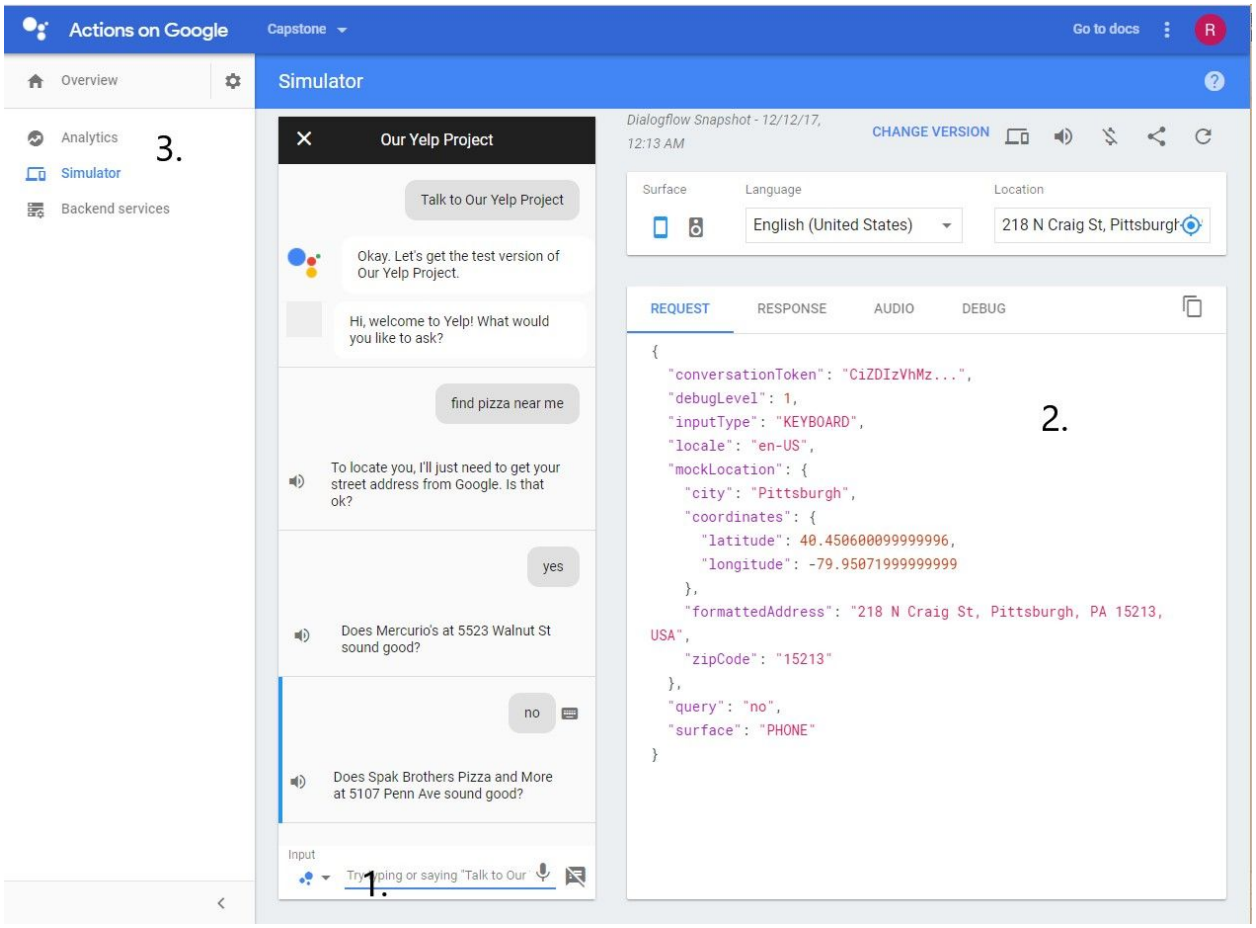
Dialogflow Rundown: Integrations



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



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**.

Firebase Rundown:

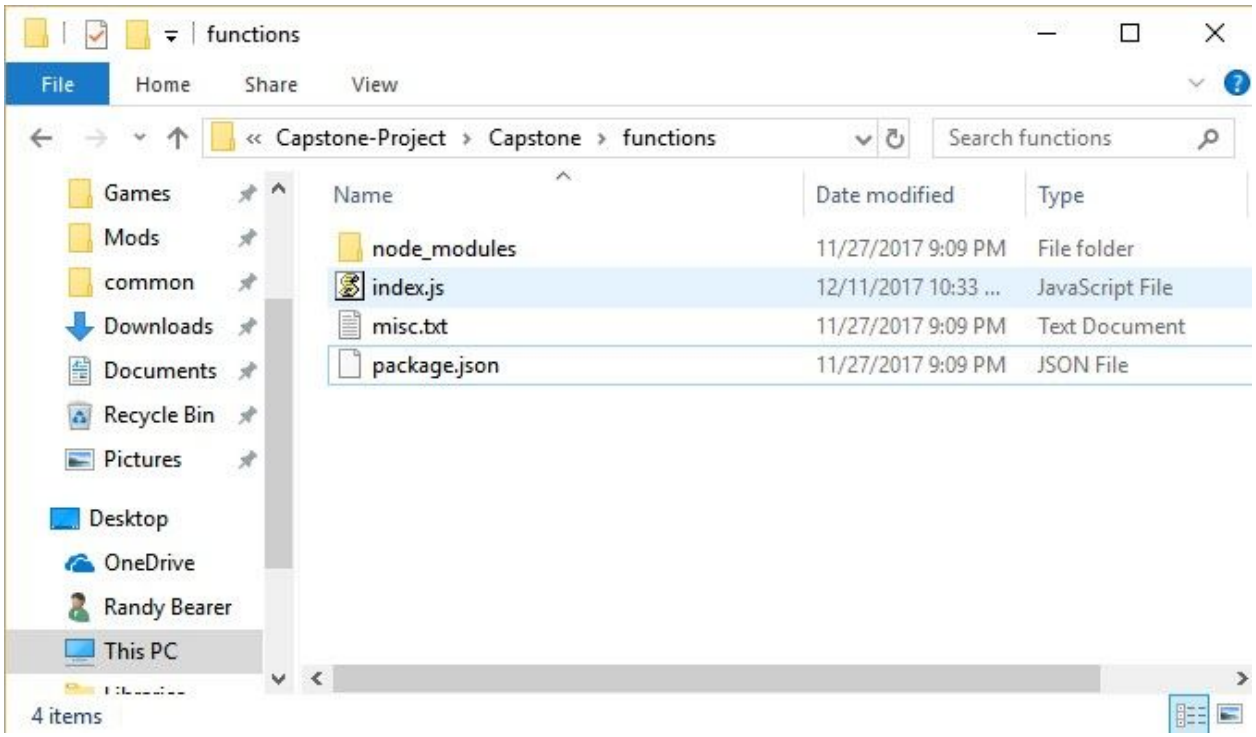
- To initialize **Firebase** for a directory, please refer to the first link under **Helpful Links** For a tutorial on getting the deployment service set up.

- 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".

    - **Firebase** 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 **Fullfilment** tab in the **Dialogflow** sidebar.

- To debug console output from the **Firebase** 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.