readme.md

Technical Description - Hazel

Hazel repository on Github

Abstract

The idea behind Hazel is to become a intelligent centralized health care information service. We want it to be able to answer any health care related question with either an educated answer or direction to the right service the patient should use.
For the prototype we decided to implement an AI personal assistant which will be integrated with one of the available personal assistants and a website for hosting rich content pages where the

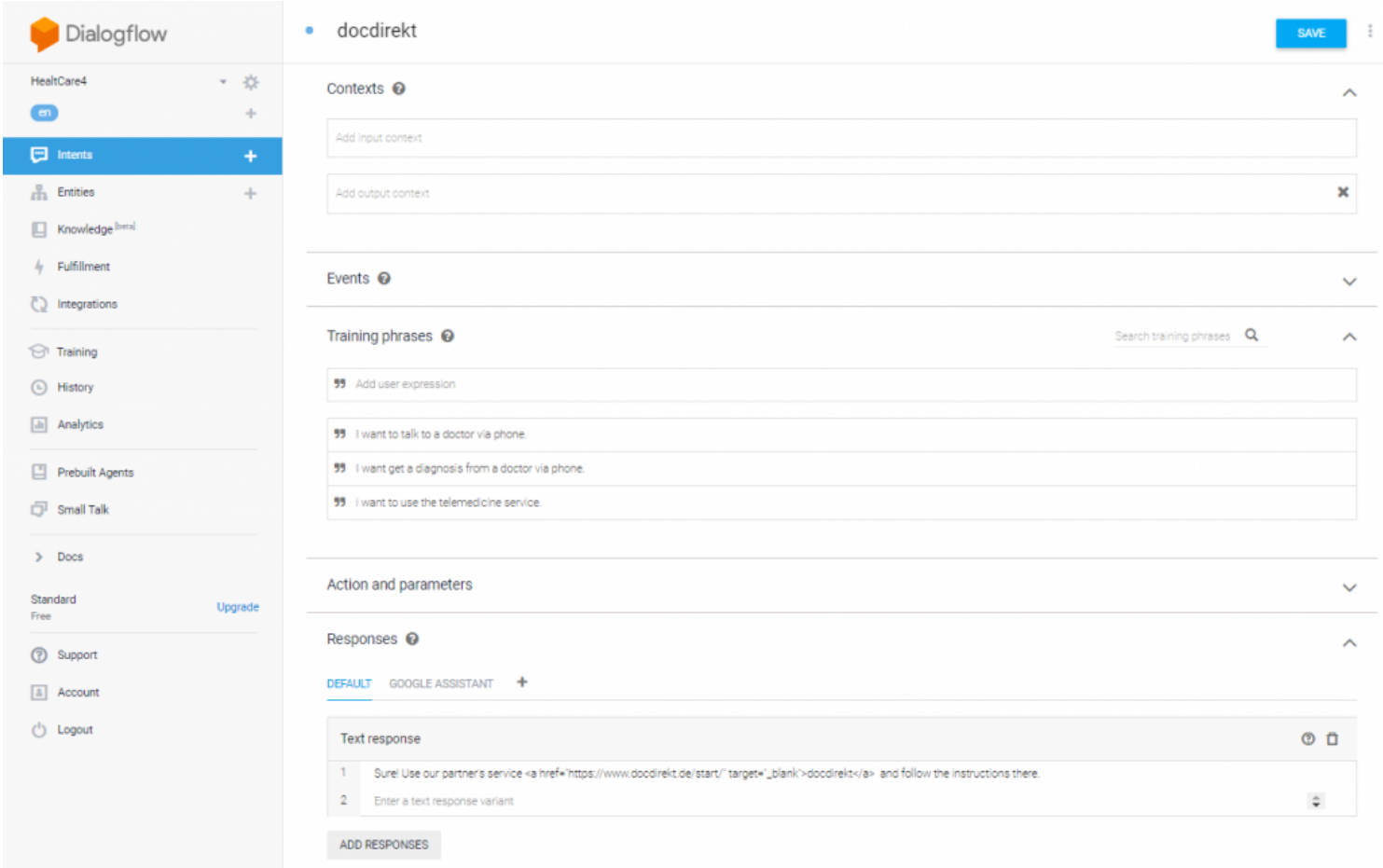
Prerequisites

- AWS Account
  - EC2 Machine with enough computing power to hold the server
  - Static IP
- Google developer account with the following services available
  - Google Dialogflow
  - Google Actions
- Freenom account with domain reserved

Personal Assistant

Dialogflow

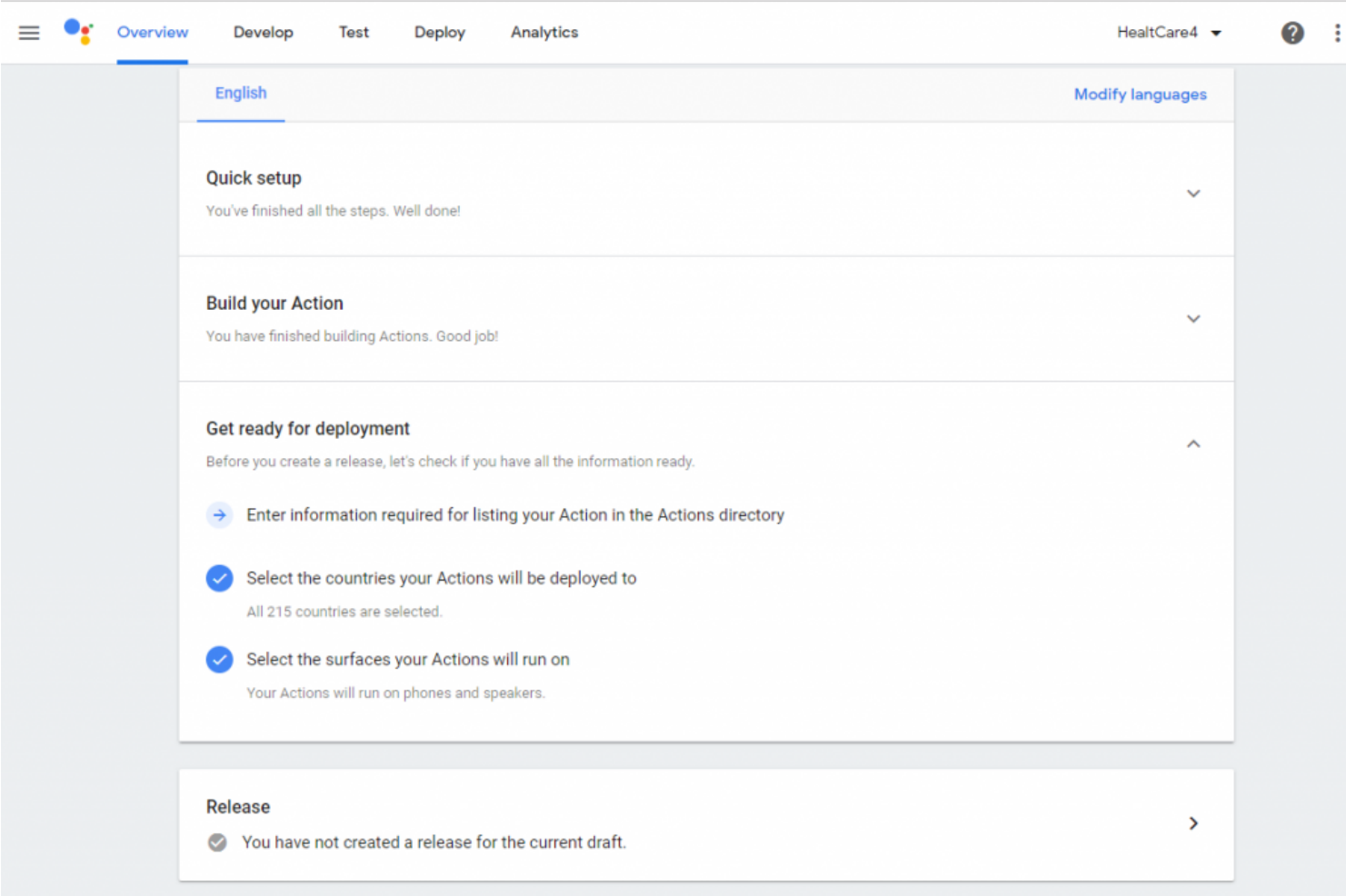
The personal assistant is implemented using Dialogflow. The dialogflow model can be trained from the Dialogflow console, where the training is done by using intents, contexts and responses as can be seen on the following screenshot:



The training files and webhooks we used can be found on the dialogflow folder.
The assistant was manually trained to support limited set of scenarios, links and interaction with user was splitted to HTML based responses which are used in the website integration, and to card and other special Google Assistant responses which are used in the Google Integration responses. Where more logic needed to be implemented in response we used webhooks which are hosted directly on our Dialogflow account.

Google Assistant Integration

For the prototype, we choose to integrate it with Google Assistant since it seems to be the most accessible and familiar.
The way to integrate application with Google Assistant is to use the Google Action Console, which let define an "Action" which is an extra capabilities set for the Google Assistant.



Using the console we have created an Action project from our dialogflow model. The console is where all information and configuration needed for deployment is found. We have trained a trigger event from the Action console for 'Please Talk to Hazel Help' and we launch our test version of the Google Assistant so it can be used from any user we have attached to the test version. After attaching our usernames on the console we could test our application on our phones, tablets or smart-home device.

Website Integration

The website integration is done using 'My Chatbot' Wordpress extension, the extension lets us easily implement a view which connects to dialogflow API using a shortcode within the Wordpress pages.

Website

The website is hosted on EC2 AWS instance with route 53 static IP (52.29.18.178) The machine is running Ubuntu 18.04

After launching the instance, Wordpress + Apache server need to be installed see manual. As part of the installation MySQL server would also be installed on the machine.

The website is using the following Wordpress plugins

- Elementor

- Insert Headers and Footers, and Scripts n Styles for adding custom javascripts
- My Chatbot to integrate our dialogflow model in the website
- Participants Database for viewing databases tables on pages
- Really Simple SSL
- Super Progressive Web Apps and AMP to support PWA (website as an app)
- wpDiscuz for improving the commenting system and add rating
- RumbleTalk Chat which enables users to chat with each other
- TablePress for creating tables
- RumbleTalk for adding chat rooms

The domain is served using Freenom, after buying static IP from AWS route 53 and attaching our server to it, on the Freenom console we have registered the static IP on 'A' record for our domain DNS.
For the SSL certificate we used Let's Encrypt, after getting the certificate we have also registered it on the server and on 'TXT' records on the DNS server.

Example pages was created for number of doctors which can be redirected from the personal assistant, we used Elementor template for easily creating multiple pages.

For showing the waiting-times there's an MySQL server running on our server which holds the following table:

> Name [Text-line], Hospital [Checkbox], Waiting Time [Numeric], Address [Text-line], City[Text-line], State [Text-line], Country [Text-line], Zip [CodeText-line], Phone [Text-line]

We have populated the table with example facilities from Munich.
We have created dedicated page which has the view of the facilities database, with option to view/hide hospitals, and to find specific facility by location (using shortcode of Participants Database extension and javascript). The waiting times are updated using a cron job on the server that is running a python script periodically which updates the waiting time - this should be the point of interface for getting the waiting times.

Scripts

We decided to use Python as our scripting environment on the server. the different scripts which were used to automate some of the tasks and to run on the servers are on the scripts folder.