

Programmer Guide

Version 1.0

Presented by

Stefon Williams

Caleb Crickette

Thomas Barton

Eugene Kim

Nicholas Ballo

Abdul Kamara

Sohail Sobhani

Anthony Cedeno

Geoffrey Dean

Jean-nae Dedrick

SWEN 670

University of Maryland Global Campus

Project Plan Approvals

|  |  |  |
| --- | --- | --- |
| Name | Signature | Date |
| Approved by:  Dr. Mir Assadullah |  |  |
| Approved by:  "Stakeholder" |  |  |
| Approved by:  "Project Manager" |  |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Name | Description of Change(s) |
| 1.0 | 10/20/2020 |  | Initial Release |
| 1.1 | 03/15/2021 | Eugene Kim | Formatting to make sure headings and TOC are good |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Introduction 4](#_Toc66739856)

[1.1. Purpose 4](#_Toc66739857)

[1.2. Intended Audience 4](#_Toc66739858)

[2. System Architecture 6](#_Toc66739859)

[2.1. Architectural Design 6](#_Toc66739860)

[2.2. Architectural Overview 6](#_Toc66739861)

[3. Application Setup 7](#_Toc66739862)

[3.1. Git and Code Deployment 7](#_Toc66739863)

[3.1.1. Downloading Git locally 7](#_Toc66739864)

[3.1.2. Downloading the Latest Code 7](#_Toc66739865)

[3.1.3. Downloading Visual Studio Code 7](#_Toc66739866)

[3.1.4. Creating Google Dialogflow Developer Account 7](#_Toc66739867)

[3.2. GCP Dialogflow and Cloud Platform 8](#_Toc66739868)

[3.2.1. GCP Dialogflow setup 8](#_Toc66739869)

[3.2.2. Dialogflow Dashboard 8](#_Toc66739870)

[3.3. Google Cloud Platform 9](#_Toc66739871)

[3.3.1. Platform Tools 9](#_Toc66739872)

[3.6. Google Dialogflow Overview 9](#_Toc66739873)

[3.6.1. Intents 9](#_Toc66739874)

[3.6.2. Dialogflow Webhook 10](#_Toc66739875)

[3.6.3. Training 11](#_Toc66739876)

[3.6.4. Other Dialogflow Functions 11](#_Toc66739877)

[3.7. Code Structure 11](#_Toc66739878)

[3.7.1. Source code 11](#_Toc66739879)

[3.7.2. Go File Descriptions 12](#_Toc66739880)

[4. Running the Application 12](#_Toc66739881)

[4.1. Local Git Repo Configuration 12](#_Toc66739882)

[4.2. Installing Go 12](#_Toc66739883)

[4.3. Running the Webhook API 13](#_Toc66739884)

[4.4. Dialogflow Instance 13](#_Toc66739885)

[5. Challenges/Issues/Concern 13](#_Toc66739886)

[6. Additional Help 13](#_Toc66739887)

[7. License Information 13](#_Toc66739888)

[Appendix A: Abbreviations and Acronyms 15](#_Toc66739889)

[Appendix B: References 16](#_Toc66739890)

# Introduction

## Purpose

This document contains technical details about the development and data aspects of the Form Scriber application, which shall be referred to as the application. Software architecture and technical details of the system needs are presented. Information on the architecture is presented to benefit those who desire to build on and create variants of the Form Scriber application.

## Intended Audience

The reader is expected to be software developer or other IT personnel that is seeking to understand or improve this application. The backend architecture, Google Dialogflow service, mobile backend and mobile interface specifics are covered.

This table covers key stakeholders that are part of this project as of writing this document.

Table 1 Technical Project Stakeholders

|  |  |  |
| --- | --- | --- |
| Name | E-mail address | Role |
| Professor Mir Assadullah | [mir.assadullah@faculty.umgc.edu](mailto:mir.assadullah@faculty.umgc.edu) | Stakeholder |
| Roy Gordon | [uspsrgordon@aol.com](mailto:uspsrgordon@aol.com) | PM Mentor |
| Johnny Lockhart | [janthony@umgc.dev](mailto:janthony@umgc.dev) | Dev Mentor |
| Stefon Williams | [swilliams538@student.umgc.edu](mailto:swilliams538@student.umgc.edu) | Project Manager |
| Caleb Crickette | [ccrickette@@student.umgc.edu](mailto:ccrickette@@student.umgc.edu) | Lead Developer |
| Thomas Barton | [tbarton7@student.umgc.edu](mailto:tbarton7@student.umgc.edu) | Developer |
| Sohail Sobhani | [ssobhani@student.umgc.edu](mailto:ssobhani@student.umgc.edu) | Developer |
| Abdul Kamara | [akamara35@student.umgc.edu](mailto:akamara35@student.umgc.edu) | Developer |
| Eugene Kim | [ekim55@student.umgc.edu](mailto:ekim55@student.umgc.edu) | Business Analyst |
| Nicholas Ballo | [nbsockem@gmail.com](mailto:nbsockem@gmail.com) | Test Engineer |
| Jean-nae’ Dedrick | [jcdedrick@gmail.com](mailto:jcdedrick@gmail.com) | Project Manager |
| Geoff Dean | [gdean5@student.umgc.edu](mailto:gdean5@student.umgc.edu) | Developer |
| Anthony Cedeno | [anthony.cedeno@gmail.com](mailto:anthony.cedeno@gmail.com) | Test Engineer |
| Ivy Pham | [ipham@student.umgc.edu](mailto:ipham@student.umgc.edu) | DevSecOps Engineer |
| Vincent Leung | [vleung1@studeng.umgc.edu](mailto:vleung1@studeng.umgc.edu) | DevSecOps Engineer |

# System Architecture

This section illustrates the design for this project. The various components of the architecture are detailed throughout, as well as the reasoning as to why the technologies used were selected. Google Dialogflow was selected because of its online availability and free cost to develop, as well as its cheap cost to test, including a free trial and $300 credit. Go is used for the webhook portion of the project due to its intuitiveness and high compatibility with other Google applications. Javascript and HTML are also used to run and display the Form Scriber website.

## Architectural Design

The Google Dialogflow service is a key component in the function of the Form Scriber application backend, while a custom made mobile front end shall be used to allow the end user professionals to see what is being recorded as well as initiating a recording session. Google Docs is used for the creation of templates and management of completed documents.

## Architectural Overview

The UI of the Form Scriber application is the mobile interface which functions on a professional's mobile device. The backend logic consists of the Google Dialogflow service, which functions in tandem with webhooks and go files to create the overall logic that drives the Form Scriber application and its ability to make dynamic intents. The webhook is sent a specific document within GCP that then triggers the document to be interpreted for fields that need to have intents created for them, and creates those intents in Dialogflow. When the user then triggers an intent, the Dialogflow service passes the data to the webhook, which then updates the document in Google Docs.

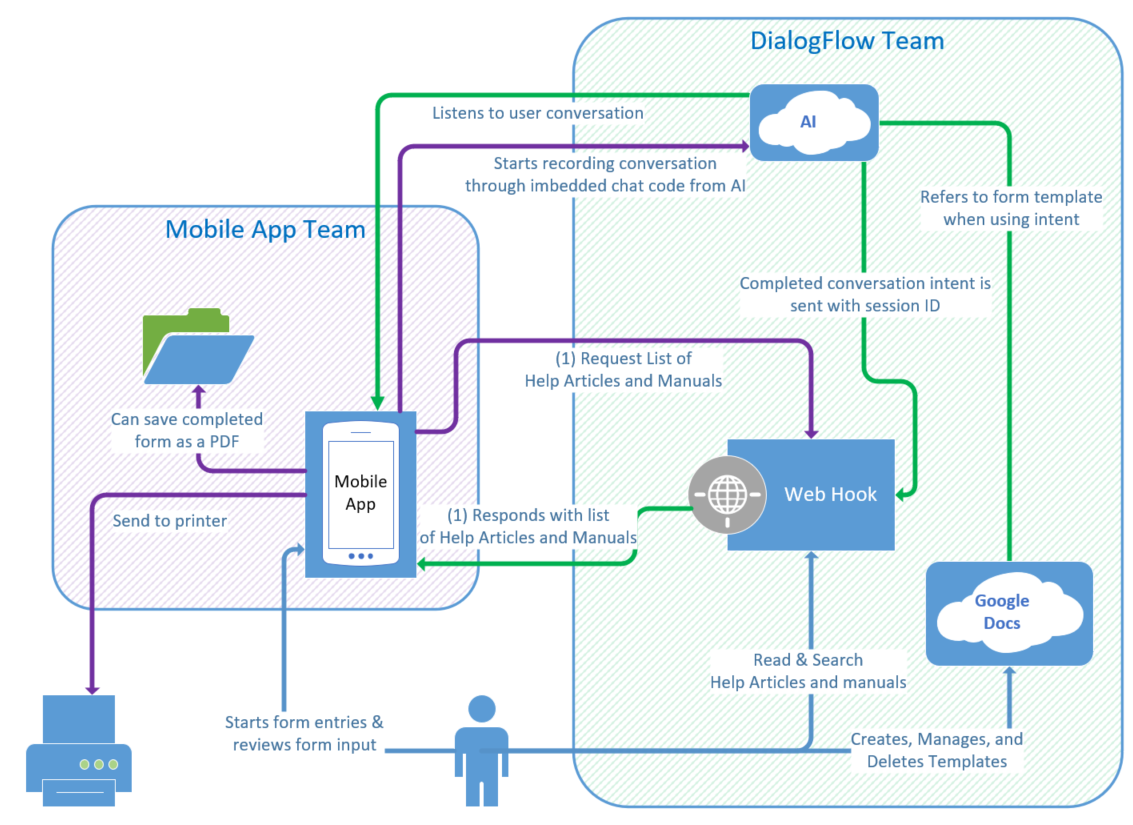


Figure 1 - System Architecture Diagram

# Application Setup

## Git and Code Deployment

The popular open access code repository GitHub is being used to host the code for the project, as well as using Git’s powerful branch management for version control. The project board in Github is also being utilized to manage project tasks and track each tasks progress. The repository being hosted online makes it easy to access for the team members as well as accessible for the purpose of the project being distributed as an open-source application. In order to access the GitHub repository, a GitHub account is required.

### Downloading Git locally

It is highly recommended to download some local git management tool to be able to pull and commit code that is modified on a local machine. For this guide, downloading Git for a local machine at <https://git-scm.com/downloads> will allow for the usage of git, please do so for the appropriate operating system.

### Downloading the Latest Code

There are multiple ways to do this, and the way to do this via command line is covered. First after installing git, open up GitHub and go to the Form Scriber repository, <https://github.com/orgs/umgc/teams/form-bot-dialogflow> and copy down the branch URL. After copying the branch URL down, use the local git to make a local git repo using ‘git checkout’, credentials will be requested when the command is executed.

### Downloading Visual Studio Code

Visual Studio Code is a freeware source-code editor that was used to develop the code. It can be downloaded here: <https://code.visualstudio.com/download>.

After downloading and installing Visual Studio Code, simply open the file that the Git repository was created in to view and edit the code.

### Creating Google Dialogflow Developer Account

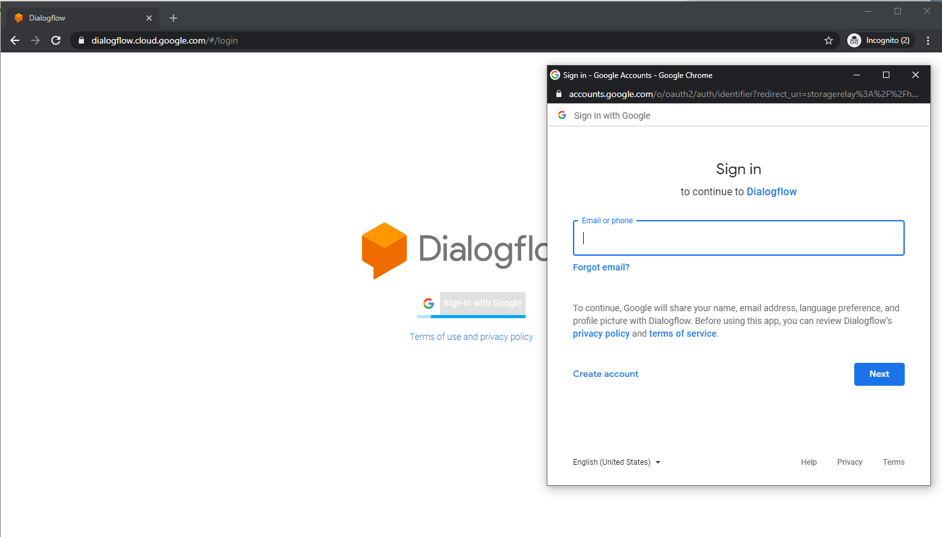
The Google Dialogflow product page can be found here: [https://cloud.Google.com/dialogflow](https://cloud.google.com/dialogflow). A $300 credit is provided for first signing up, which will require a Google account as well as some form of payment information (as a means of reducing the ability to simply make a second account as easily).

## GCP Dialogflow and Cloud Platform

### GCP Dialogflow setup

After getting into the Dialogflow console the project can be imported from a zip file, which is provided within the Git repository. Simply click on the gear next to the dialogflow project name and then click on export and import, then say restore from zip, choosing the zip file mentioned earlier.

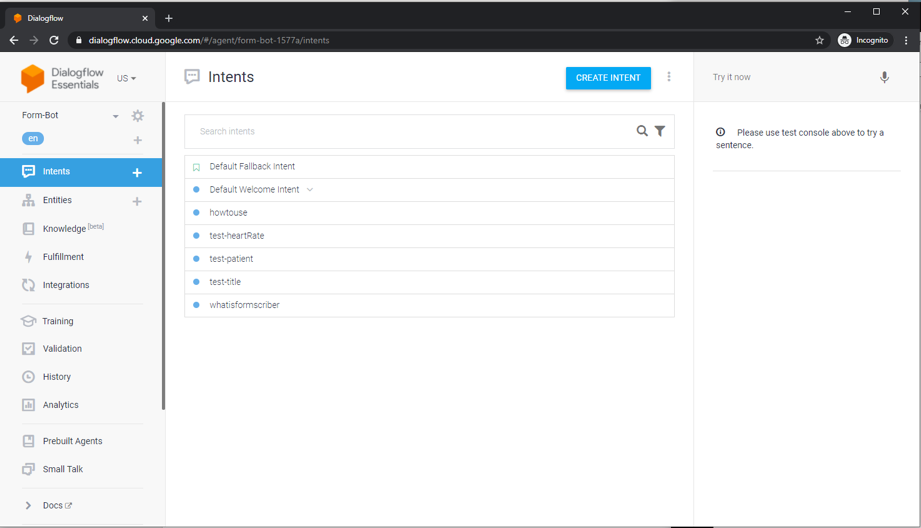
Figure 2 Google Dialogflow Login



### Dialogflow Dashboard

The dashboard for Dialogflow allows for the customization of the chat agent, with additional inputs being possible to create for further user interactivity.

Figure 3 Google Dialogflow Dashboard



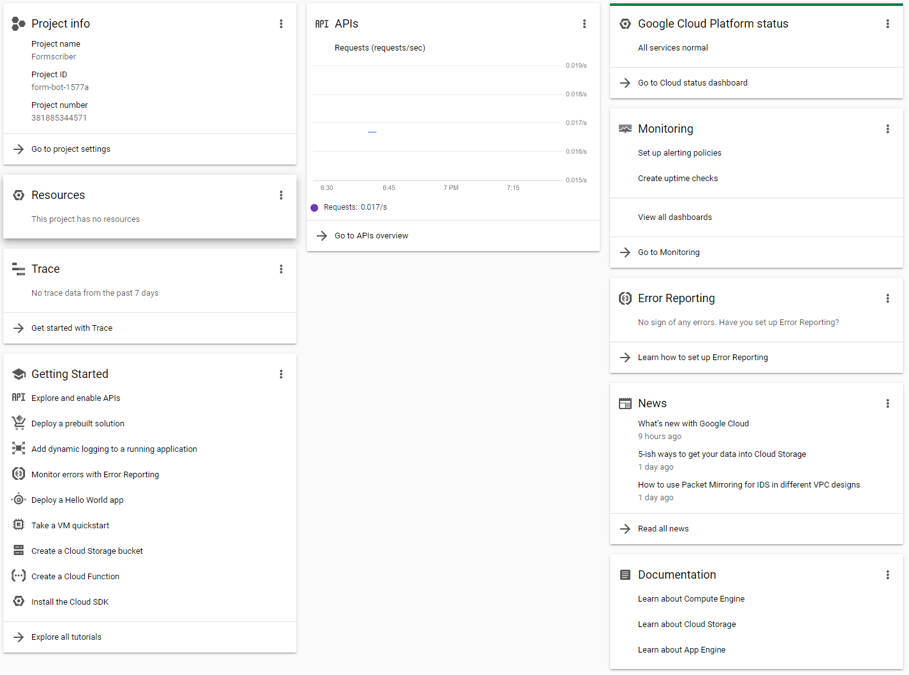
## Google Cloud Platform

This platform is what is used to manage all the billing and permission portions of the project. Form here users can be added or taken off the project and billing adjusted if expected that the implementing users wish to continue to use the project and use the interpretation of intents that require per interpretation billing.

### Platform Tools

There are many tools that are able to be accessed through the Google cloud platform. Various APIs and documentations, as well as metrics for API usage and other monitoring tools.

Figure 4 Google Cloud platform tools



## Google Dialogflow Overview

Google dialogflow is the main driving factor of the entire application that allows for the creation and management of intents, as well as the training of those intents. The intents are able to take input and extract the data, which then gets passed to the webhook to get put into the appropriate form fields. Refer to figure 2 for the general screen of the Google Dialogflow Console.5

### Intents

The intents are the individual interpretation units that determine what overall input the user is intending to create. The default welcome intent is the main A default welcome intent is used to tell the user that the Dialogflow agent is working, and then from there the specific documents URL is captured. Capturing the URL triggers parsing the document for fields which then generate intents that the user then can use to fill in the various form fields.

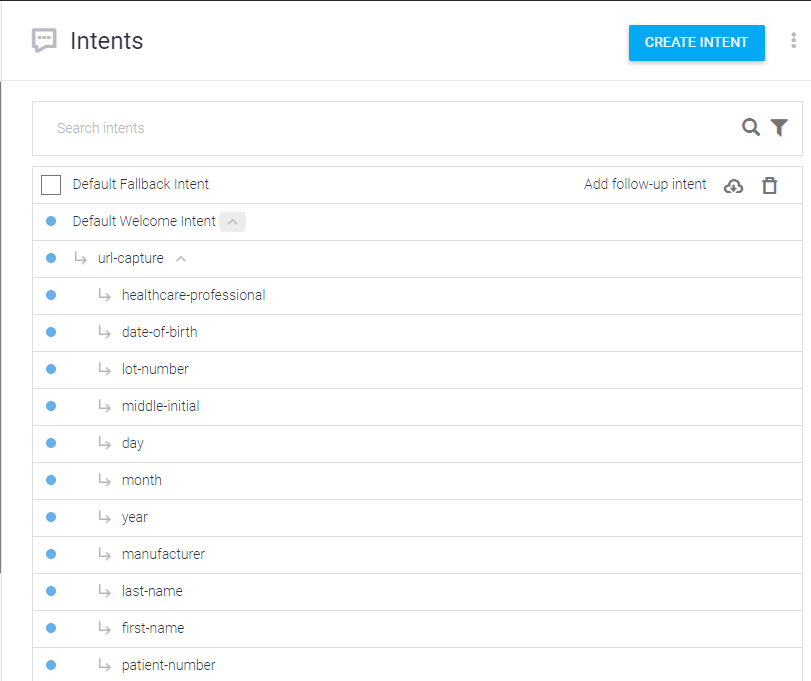


Figure 6 List of User's Dialogflow Intents

### Dialogflow Webhook

The webhook in perspective from the Google Dialogflow Service is any web endpoint that Dialogflow must interact with, and will initiate a POST to that web endpoint via a post. In the case of the Form Scriber application, the endpoint the Dialogflow service interacts with is a custom webhook API that retrieves information from forms that the user specifies, as well as takes the user input interpreted from dialogflow and inserts them into the document templates.

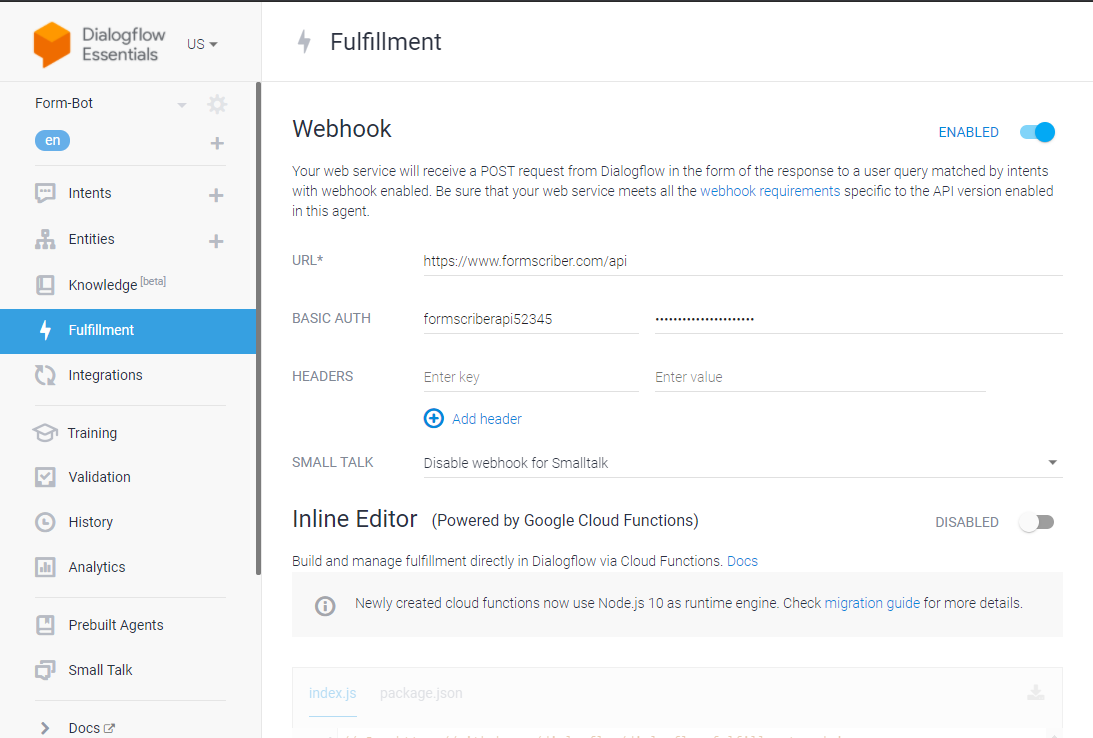


Figure 7 Dialogflow webhook fulfillment page

### Training

This training page gives statistics on what phrases get input into dialogflow, and how many times they appear. When the phrase was used as well as the times when it did not match an intent are shown.

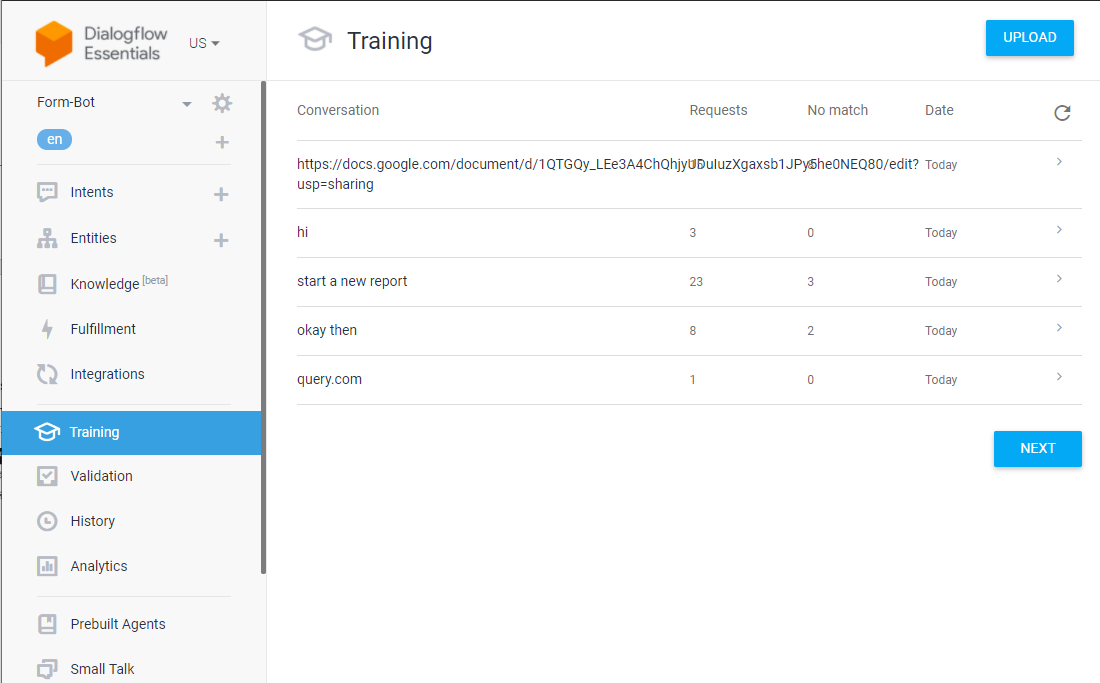


Figure 8 Dialogflow training page

### Other Dialogflow Functions

There are several other Dialogflow functions that were not used during the course of development, and therefore are not detailed within this document.

## Code Structure

This projects webhook is programmed with Go, a language that was made by Google and is used due to its high compatibility with the Google products being used as well. The mobile interface will integrate Google Assistant with the use of flicker to add some other functionality. In addition, Go was used for its superior backend system programmability when it comes to web services. Also, Go has concurrency built in which is important for asynchronous services and to be as performant as possible.

### Source code

The source code for this project is comprised of the Go, Vue and other files located within the Form Bot repository. The main components are as follows:

* The Go webhook files.
* The Form Scriber Google Dialogflow project

### Go File Descriptions

|  |  |
| --- | --- |
| Filename | Description |
| main.go | This is the main driving for the Form Scriber webhook. The other go files get called form here. The API endpoints are defined here in detail with paths that allow for some limited input via the post calls. It responds to calls using json. In addition, there is some logging functionality. |
| Get\_document.go | Get\_document takes an entire Google doc that is appropriately marked with field brackets and interprets it and more importantly, what fields are in the document get read in and put in as intents into the Dialogflow service. |
| GetToken.go | This is a token fetching class for authenticating with Google services. |
| Logger.go | This is a simple logging class that writes out to the log file when the application started, and messages for what gets logged from the other classes. |
| ReadConfig.go | Reads the config files that are present to set up the webhooks upon loading the application. |
| String\_Functions.go | Converts various string inputs for formatting reasons. |
| Update\_document.go | This is the class that takes the inputs from Dialogflow and replaces the fields in the document with those inputs. |
| Webhook.go | This is the official webhook portion that handles all the gets and posts, forming the web API and allowing for JSON post calls to this service. |
| Dynamic\_Intent.go | This file contains the function for creating the generic dynamic intents in Dialogflow utilizing Dialogflow API’s for batch intent creation based on the Google doc. |

# Running the Application

Now each of the individual configurations to get the application running will be explained.

## Git Repo Configuration

To review, the local git repository should be created through git bash or another equivalent git account by checking out the code from the remote repository. The repository can be cloned at the leisure of those that desire to, after which it is advised that the overall flow of the branches be that there is a stable master branch that allows for integration between the Dialogflow and UI portions of the project. Beyond that, for the current mobile and Dialogflow teams commit versions of their code to specific development branches which then get merged into master.

Those that wish to make modifications can NOT commit to the original Github repository but must instead fork the Github repository and maintain their own fork.

For forking the Dialogflow webhook Github repository to customize it further, see this article about forking a Git repository. https://docs.github.com/en/github/getting-started-with-github/fork-a-repo

## Installing Go

In order to run the webhook, as well as being able to properly compile and parse the webhook Go must be installed. The Webhook was written with Go version 16.0, however future Go versions should be acceptable. Go can be downloaded here: [**https://golang.org/**](https://golang.org/)

## Running the Webhook API

After installing Go, put in the command line go build where main.go is. An executable should be formed, running this executable will create an instance of the Webhook API that runs natively. With this said, it is recommended to set up a Kubernetes instance to have the webhook be accessible from there.

## Dialogflow Instance with kuberneties

A local system-based instance of Dialogflow will need to be created per organization by standing up a Kubernetes instance of the Dialogflow service. Detailed instructions on how to do this are within the Form Scriber Deployment guide.

The code for this project is stored on the UMGC SWEN 670 GitHub repository. That repositories can be found here: <https://github.com/umgc/umgc.formbot.dialogflow>

# Challenges/Issues/Concern

Form Scriber has dependencies as well as room for improvement that can further enhance the function of the application to the needs of future developers. These improvements will require many hours of work.

* High dependency on voice recognition renders Form Scriber susceptible to errors due to misunderstanding and incorrect intent recognition. A possible remedy would be to use Google Assistant to take advantage of the personalized voice recognition feature, which Dialogflow messenger’s do not possess. This would allow learning and adapting the specific users voice, which could potentially increase quality.
* Simulated and predicted situations render the intents used for dialogflow at a bare bones state, leaving much room to create additional intends to further enhance the users experience by making the application have better flow.

# Additional Help

If additional help is required, first contact the UMGC Masters in Software Engineering department before reaching out to the team members listed in this document.

# License Information

Visual studios code is a free development IDE developed by Microsoft. Any instance of Visual Studio code is only licened and not sold. Code for the IDE is available under the MIT licensing agreement at [**https://github.com/Microsoft/vscode**](https://github.com/Microsoft/vscode).

Initial usage of Google Dialogflow is subject to the license agreement of the trial software. Upon agreeing to pay for the software a different term of service goes into effect. The terms of service are here: https://cloud.Google.com/dialogflow/docs/terms-trial-edition

The Kubernetes deployment package and application are open-sourced projects managed by an Apache 2.0 licenses.

All software developed under this project is deemed to be open-source educational work and available for use and modification. Please credit this team when creating and distributing modifed versions of this tool, or when using any of the techniques detailed within.

The SWEN 670 team will NOT be responsible for setting up new individual instances for other users or parties. Any other user or party that wishes to use the Form Scriber tool may do so, but assumes all responsibilities for hosting the application on their servers, registering the service with Google, and maintaining the application instance that they download and initiate.

1. Abbreviations and Acronyms

|  |  |
| --- | --- |
| Acronym/Abbreviation | Definition/References |
| AI | Artificial Intelligence – An application that aims to mimic human intelligence. |
| API | Application Programming Interface |
| DevSecOps | Development, Security and Operations – Group of developers responsible for the deployment and security of an application. |
| IDE | Integrated Development Environment |
| REST API | Representation State Transfer – An API for interacting with data. |
| SRS | Software Requirements Specification |
| UI | User Interface – The part of the application that users use to interact with the application. |
| URL | Uniform Resource Locator |
| GCP | Google Cloud Platform |

1. References

Go (n. d.) Go (download and install website) <https://golang.org/>

Github (n. d.) Kubernetes / LICENSE <https://github.com/kubernetes/kubernetes/blob/master/LICENSE>

Google (n. d.) Dialogfow Trial Edition terms of service https://cloud.Google.com/dialogflow/docs/terms-trial-edition