

Technical Design Document

Version 1.3

Presented by

Michael Le

DJ changing

Thomas Barton

Eugene Kim

Nicholas Ballo

Abdul Kamara

Sohail Sobhani

SWEN 670

University of Maryland Global Campus

Technical Design Document Approvals

|  |  |  |
| --- | --- | --- |
| Name | Signature | Date |
| Approved by:  Dr. Mir Assadullah |  |  |
| Approved by:  "Stakeholder" |  |  |
| Approved by:  "Project Manager" | Stefon Williams | 2/24/2021 |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Name | Description of Change(s) |
| 1.0 | 2/1/2021 | Thomas Barton | Initial Release |
| 1.1 | 2/13/2021 | Eugene Kim | Formatting and adding Sec. 7 |
| 1.2 | 2/21/2021 | Eugene Kim | Updating requirements matrix and name due to GCP policies. |
| 1.3 | 2/22/2021 | Stefon Williams | Editing and corrections |

Table of Contents

[1. Introduction 4](#_Toc65082104)

[1.1 Purpose 4](#_Toc65082105)

[1.2 Scope 4](#_Toc65082106)

[1.3 Overview 4](#_Toc65082107)

[1.4 Reference Material 4](#_Toc65082108)

[1.5 Definitions and Acronyms 5](#_Toc65082109)

[2. System Overview 6](#_Toc65082110)

[3. System Architecture 7](#_Toc65082111)

[3.1 Architectural Design 7](#_Toc65082112)

[3.2 Decomposition Description 8](#_Toc65082113)

[3.2.1 Dialogflow Form Bot Agent 8](#_Toc65082114)

[3.2.2 Web Application Layer 8](#_Toc65082115)

[3.2.3 Webhooks 8](#_Toc65082116)

[3.2.4 Database 9](#_Toc65082117)

[3.3 Exception Handling 9](#_Toc65082118)

[3.4 Design Rationale 9](#_Toc65082119)

[3.4.1 Dialogflow Form Bot Agent 9](#_Toc65082120)

[3.4.2 Web Application Layer 9](#_Toc65082121)

[3.4.3 Webhooks 9](#_Toc65082122)

[3.4.4 Database 9](#_Toc65082123)

[4. Data Design 11](#_Toc65082124)

[4.1 Data Description 11](#_Toc65082125)

[4.2 Data Dictionary 11](#_Toc65082126)

[5. Component Design 13](#_Toc65082127)

[5.1 Service Layer 14](#_Toc65082128)

[5.2 REST Controller 14](#_Toc65082129)

[5.3 Formscriber AI Component 15](#_Toc65082130)

[6. Human Interface Design 17](#_Toc65082131)

[6.1 Overview of User Interface 17](#_Toc65082132)

[6.2 Form Bot Log In. 17](#_Toc65082133)

[6.3 Chatbot Introduction. 17](#_Toc65082134)

[6.4 Basic Functionality 17](#_Toc65082135)

[6.5 Web Based Form Management 18](#_Toc65082136)

[7. Requirement Matrix 19](#_Toc65082137)

1. Introduction

Purpose

This document details the design and architecture that shall be implemented for the Dialogflow portion of the Form Bot application. The details in the document include interfacing information and specific design decisions for the implementation of the Dialogflow service, and the interactions between the other services and the Dialogflow service.

Scope

The overall scope of the application is to allow a service professional (the user) to record specific audio that gets identified. The audio is then processed and data is extracted from it using AI. The processed data is then stored into a database to put into a report form. The report can then be printed or saved as an official record.

The Dialogflow portion of the application is responsible for setting up the analytical engine using Google’s Dialogflow service, for the overall Form Bot application as well as processing and storage of user voice data. This document details the internal architecture for the Dialogflow service, the webhooks that connect the Dialogflow service, and the integration between the web and mobile services of the Form Bot application.

Overview

The TDD has four sections that describe the major dimensions of the Dialogflow service architecture. These sections are broken down as specified below:

* **System Overview**: This is the high-level description of the system, its functionality and the decisions and contexts that the functionality is implemented within.
* **System Architecture**: This section provides a breakdown of the different portions of the Dialogflow service that must be adapted for the Form Bot application and a breakdown of the webhooks and their various pass off points to the web and mobile portions of the Form Bot application.
* **Component Design**: This section will detail the specific components that the Dialogflow team are working on and their internal design decisions.
* **Human Interface Design**: Because the final product will have a front-end design from the web and mobile teams, our team will not be focusing on any UI elements. However, we will be developing a minimal, introspective UI to demo and debug our services that will be described in this document.

Reference Material

Table 1 References

|  |  |
| --- | --- |
| Title | Reference Location |
| Course Material | UMGC SWEN 670 Course Material |
| Dialogflow Docs | <https://cloud.google.com/dialogflow/docs> |

Definitions and Acronyms

Table 2 Definitions and Acronyms

|  |  |
| --- | --- |
| Acronym/Abbreviation | Definition/Reference |
| TDD | Technical Design Document |
| GCP | Google Cloud Platform |
| UI | User Interface |
| ETL | Extract, Transform, Load in regards to data processing |
| NoSQL | Not only Structured Query Language |
| ReST | Representational State Transfer |
| HTTP | Hypertext Transfer Protocol |
| URL | Uniform Resource Locator |
| CA | Certificate Authority |
| SSL | Secure Sockets Layer |
| API | Application Programming Interface |
| AI | Artificial Intelligence |
| RDBMS | Rational Database Management System |

1. System Overview

Dialogflow is an application that interfaces between the end user (professional) and records their voice. The service does additional processing to help the end user fill out their report. The use of a mobile UI for recording, webserver for form management and creation, and Google’s Dialogflow service AI that interprets the audio shall form the final product.

At a high level, the mobile team is focusing on building out a UI that can identify the user and allow them to record their voice using an API that our team will provide. The web team is focusing on building a form creation and management platform where users can create and update forms that they have defined. The Dialogflow team will focus on building out the middleware between these two components by receiving the processing the recorded audio in real time and storing the data in a database for use on the web UI. DevSecOps will need to provide a database for all teams to use for the storage of processed data and audio records as well as create the appropriate logging tools to debug and audit system applications.

The Dialogflow service architecture will consist of four different components that are explained in further detail in the next section:

1. Dialogflow Form Bot Agent
2. Web Application layer
3. Webhooks
4. Database
5. System Architecture

This section details the different aspects of the system in a high-level context.

Architectural Design

The system is hosted on a cloud server environment. It is comprised of 4 main components: the front-end Form Bot agent UI, application layer Google Dialogflow service, webhook services, and the database. The Dialogflow agent is a virtual agent that processes natural language in text and audio format. The agent converts natural language to structured data for processing. The application layer resides within Google Cloud Services and integrates with the agent, database, and webhooks. The webhook service handles calls from the Dialogflow API Engine. The system utilizes a MySQL database engine for data storage. The webhook is a simple HTTP server with RESTful services programmed in Go.

|  |  |  |
| --- | --- | --- |
| Step | Process | Error Handling |
| 1 | Users gives URL of template to Form Bot |  |
| 2 | Webhook Parses URL and fetches all available fields from form | If error, then let end-user know |
| 3 | Webhook Backend does a BatchUpdate routine to Dialogflow create generic intents |  |
| 4 | End user starts filling out the form with commands like "Title Hello World"  Webhook parses all requests of user and make a batchUpdate to the document. | If error, then let end-user know |

Figure 1 - Process Sequence

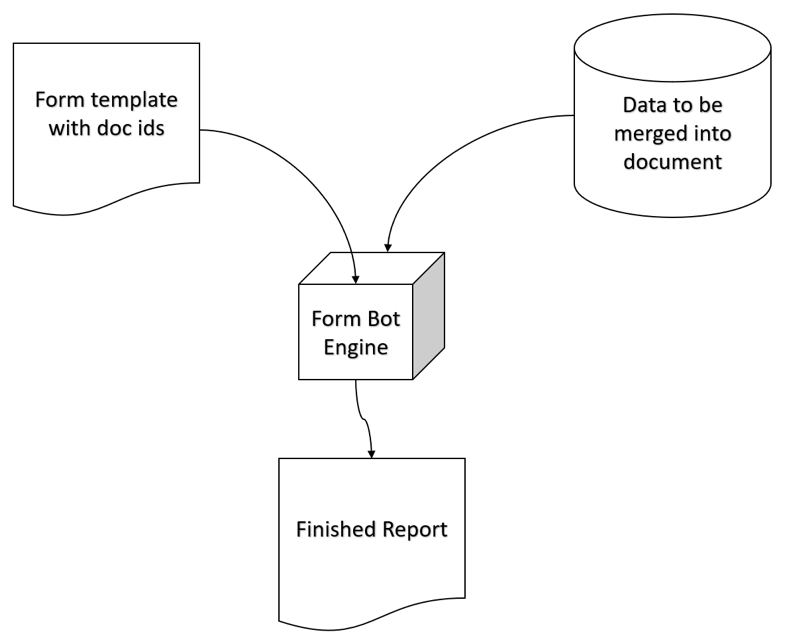


Figure 2 - Data Diagram

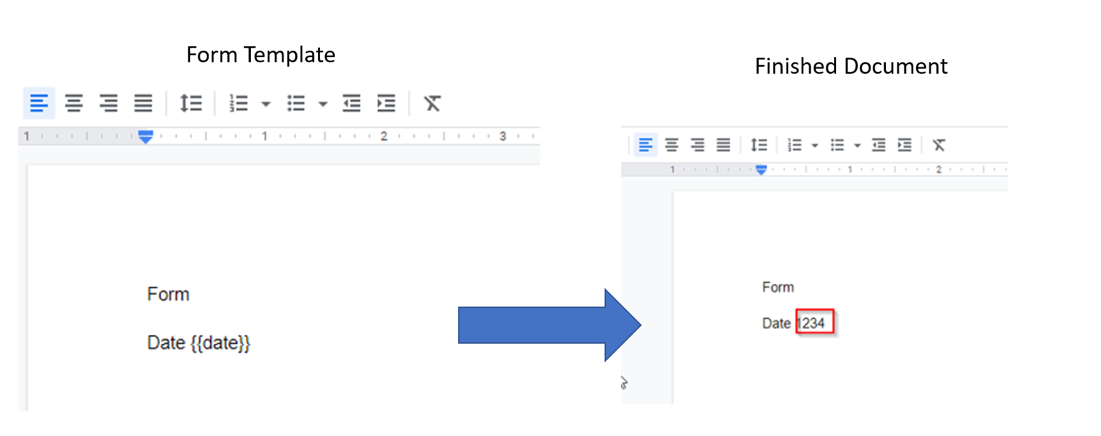


Figure 3 - Example of the process between template and finished document

Decomposition Description

Each system component fulfills a processing requirement. The basic processing requirements are to gather user input, process user input; ETL processed data, and store data. The following is a more detailed description of each component.

* + 1. Dialogflow Form Bot Agent

The agent categorizes text and audio input as intents. An intent is the end-user’s goal. It is a categorical concept allows development to translate and organize intents into entities. Each intent parameter has an entity type. Entity types dictates how data from the end user’s expression is extracted. Synonym use allows possible variations in user expressions to be categorized as a specific entity. The default agent responses are static. Fulfillment is the response provided to the end user from the agent. This application will require custom fulfillment responses calling webhook services from the database.

* + 1. Web Application Layer

The application layer is built on Go and does the data handling between the webhook, web, and mobile applications. Within the application layer are logics designed to act on various types of report data and write this data to a database.

* + 1. Webhooks

Webhook services are handlers allowing the application layer to receive real-time information. Webhook services handles calls from Google Dialogflow engine. The webhook HTTP server utilizes a mandatory CA SSL certificate in order to receive communication from Google. In addition, this service must respond to Google’s API request within 10 seconds, otherwise the Dialogflow agent will default to a standard response. The Webhook essentially will listen for the call from Google services which are controlled and created from the Dialogflow console fulfillment process. This means that the webhook should be designed in a generic way to handle intent entities which will be used for the forms later on. Due to the nature of Dialogflow being very complicated to create intents and understand which data to capture, it is necessary that all intents for the forms need to be designed in advanced in order that the webhook knows where to properly insert those values in the database.

* + 1. Database

The data resides within a relational MySQL database. MySQL is a standard relational database management system, which was chosen for several reasons. One primary reason is that it is free to use and that it is has fairly decent performance as a standard RDBMS. NoSQL was also viewed as an option, but the team had more experience with SQL databases and the advantages of NoSQL were not enough to make the move to NoSQL. Another point was that that data for the reports is fairly structured which makes sense for a typical SQL RDBMS. The database will be shared between the teams, as the web team will be writing data there for the form, Dialogflow will be writing data that the end-user speaks, and lastly the mobile app will be reading and generating the report based on this data. The database integrates with the application layer receiving calls to progress the application to fulfilling the end user’s request.

Exception Handling

Dialogflow engine handles input error handling by either going to the fallback intent or configuring a response to try to receive the input data again. Errors that occur within the ETL steps will be logged and notified to the user through the webhook to notify users to try to recapture the data point.

Design Rationale

The design rational is formed through utilizing solutions configured within the boundaries of project, technical, and functional requirements.

* + 1. Dialogflow Form Bot Agent

The Form Bot Dialogflow service will receive data from the mobile front end that will embed our chatbot agent into their app. Any input errors from the users, such as not being able to understand the user’s audio, will be relayed back to the user on the screen to try again. Due to the nature of the application’s development process the AI chatbot agent must be free, and accessible by future development teams. The agent is not open source. This allows the application to integrate the Google agent at no cost, and expands future operability and extensibility.

* + 1. Web Application Layer

The application layer utilizes Go for operability and extensibility. Go or Go lang is a statically typed, compiled programming language designed at Google from the inventors of Unix OS and C language. Go was chosen for its ability to run anywhere and be compiled directly to machine code in a fast and flexible manner. Go is fully open source and has a vast library for integrating with Dialogflow. Since this project utilizes heavy back-end services Go is the clear winner for its built-in concurrency and fast performance.

* + 1. Webhooks

Webhook services employs multiple APIs to handle call backs between the application layer and Dialogflow agent.

* + 1. Database

MySQL is a common, open source, and widely used relational database engine. The MySQL database engine provides data security, on-demand scalability, round the clock uptime, and other features. It’s wide use and extensive features will avoid future conflicts within compatibility and extensibility.

1. Data Design

### Data Description

These are the data entities that get stored in the database, which will be pulled to create the final report designed by the user.

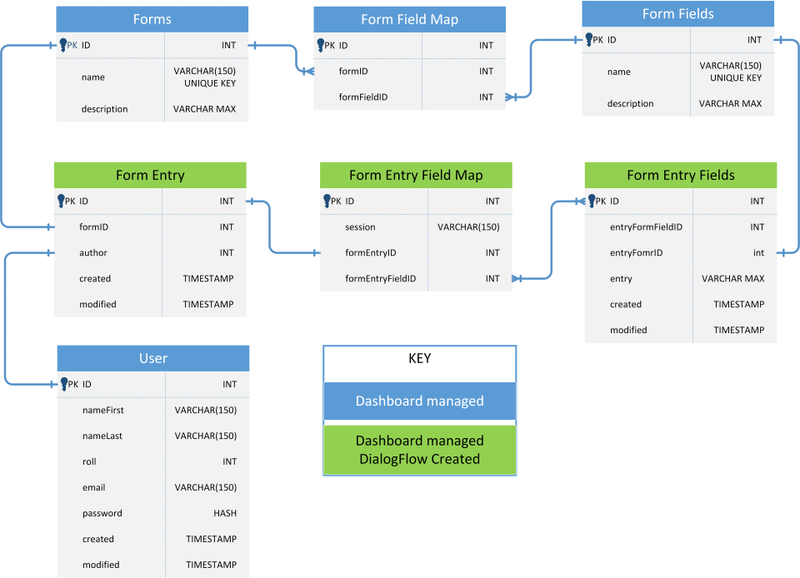
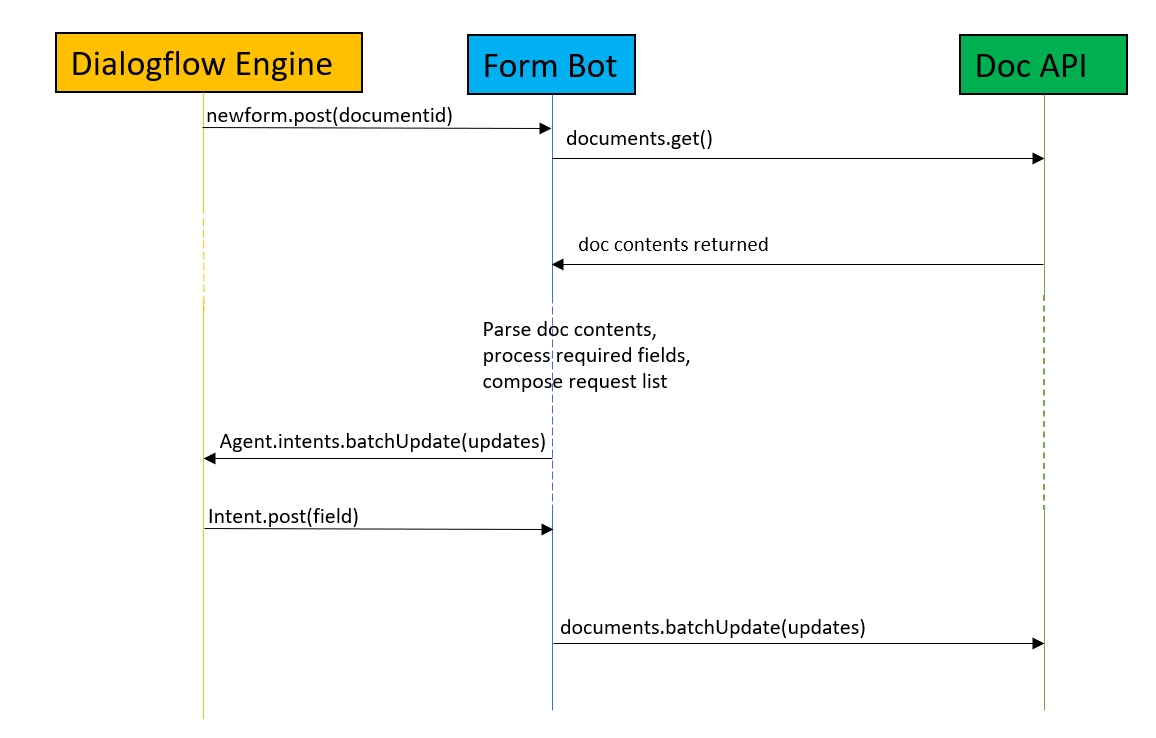


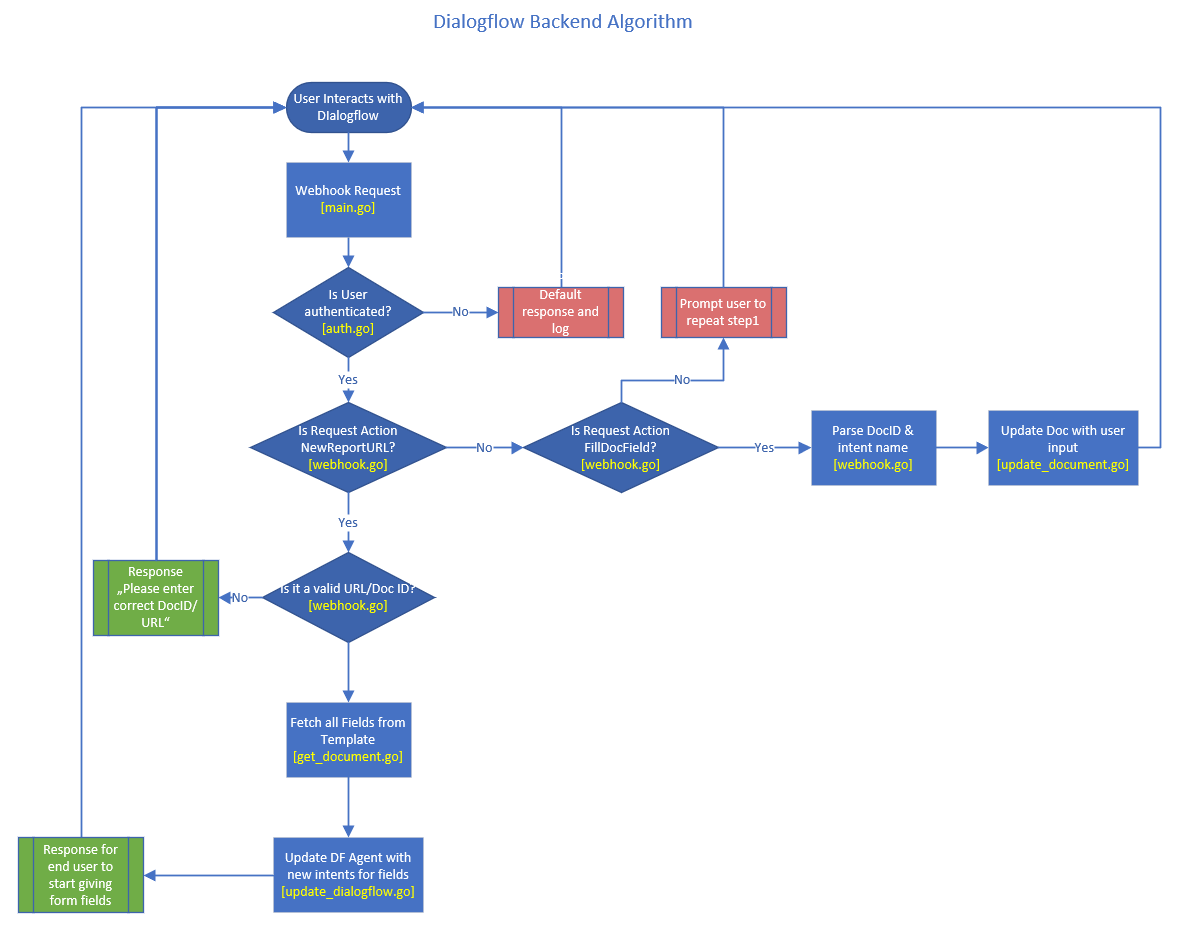
Figure 4 Database Structure Diagram

Data Dictionary

| Entity | Field | Type | NULL | Default |
| --- | --- | --- | --- | --- |
| Forms | ID | INT | No |  |
|  | name | VARCHAR(150) | No |  |
|  | description | VARCHAR MAX | No |  |
| Form Field Map | ID | INT | No |  |
|  | formID | INT | No |  |
|  | formFieldID | INT | No |  |
| Form Fields | ID | INT | No |  |
|  | name | VARCHAR(150) | No |  |
|  | description | VARCHAR MAX | No |  |
| Form Entry | ID | INT | No |  |
|  | formID | INT | No |  |
|  | author | INT | No |  |
|  | created | TIMESTAMP | No |  |
|  | modified | TIMESTAMP | Yes |  |
| Form Entry Field Map | ID | INT | No |  |
|  | session | VARCHAR(150) | No |  |
|  | formEntryID | INT | No |  |
|  | formEntryFieldID | INT | No |  |
| Form Entry Field | ID | INT | No |  |
|  | entryFormFieldID | INT | No |  |
|  | entryFormID | INT | No |  |
|  | entry | VARCHAR MAX | No |  |
|  | created | TIMESTAMP | No |  |
|  | modified | TIMESTAMP | No |  |
| User | ID | INT | No |  |
|  | nameFirst | VARCHAR(150) | No |  |
|  | nameLast | VARCHAR(150) | No |  |
|  | roll | INT | No |  |
|  | world | VARCHAR(150) | No |  |
|  | password | HASH | No |  |
|  | created | TIMESTAMP | No |  |
|  | modified | TIMESTAMP | No |  |

1. Component Design





Service Layer

The service layer resides within the application in the form of a REST controller that allows interaction between the Dialogflow service and the other portions of the Form Bot application. This allows for a layer of abstraction to occur between the different portions of the application.

REST Controller

The REST controller will use a variety of HTTP address requests that will facilitate appropriate responses and information exchange. Many of the REST controller endpoints are facilitated by the webhook portion of the application.

Table 4 REST Controller

| GET Method and Endpoint | Type | Definition |
| --- | --- | --- |
| Newform.post(documentid) | POST | Integrates with the webhook using the documentId which we need to parse and populate. |
| Documents.get() | GET | Obtain a list of fields from the template |
| Agent.intents.batchUpdate(updates) | POST | Dialogflow’s API for updating intents. |
| Intent.post(field) | POST | Webhook integration between Dialogflow to Form Bot with intent and field that needs to be updated in the form. |
| Documents.batchUpdate(updates) | POST | Updates form with values parsed from Dialogflow. |

## 5.3 Formscriber AI Component

The below flowchart represents the high-level processing of data input from mobile interface to the resultant generated form. The front-end mobile application will be presented to the user in which they input a variety of data, via voice recognition, that then gets processed by the Dialogflow service. After the Dialogflow service processes the data, it is tagged appropriately and sent to the database, where the web application will take the tagged data and put it within the appropriate sections of a form.

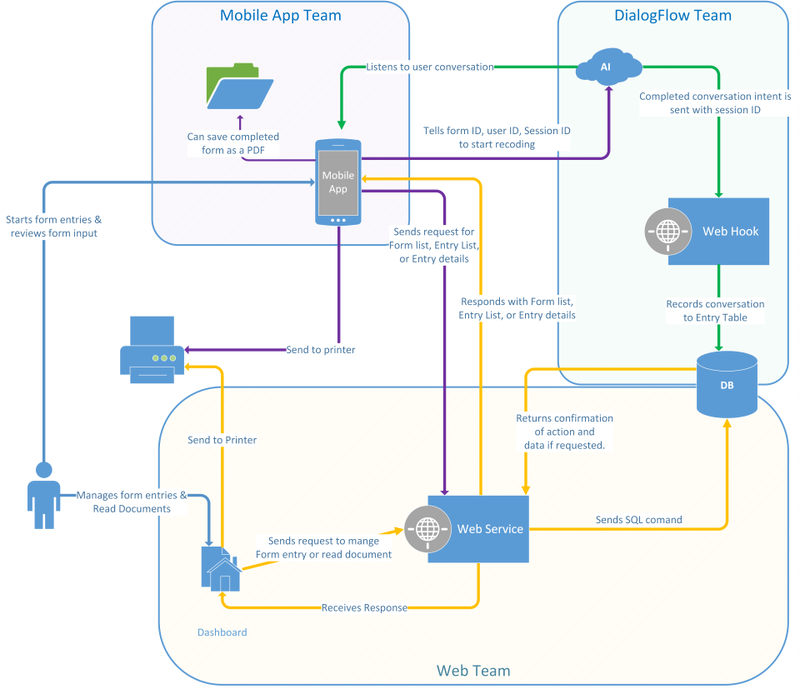


Figure 5 Formscriber AI Dialog Flow

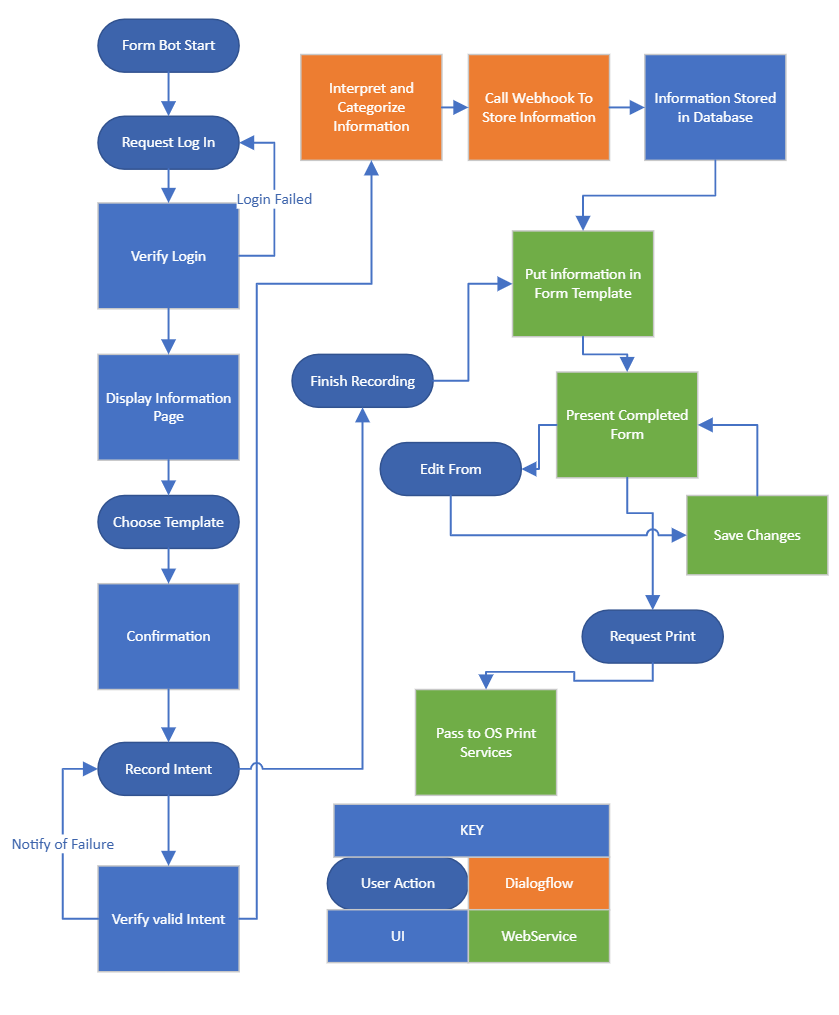


Figure 5 Formscriber AI Dataflow Diagram

1. Human Interface Design

### Overview of User Interface

This section will detail a series of user interface prototypes. These prototypes are not final; however, they will give a general idea of what sort of experience the user will have.

### Form Bot Log In.

The Form Bot mobile application will initially present a log in screen.

### Chatbot Introduction.

This part of the application is show after logging in and displays important information about form bot, such as the basics for how to use it, as well as privacy policy and terms of use.

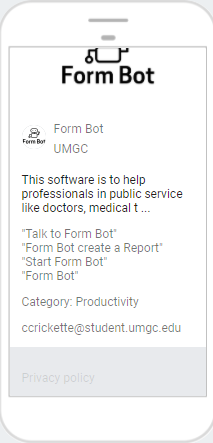
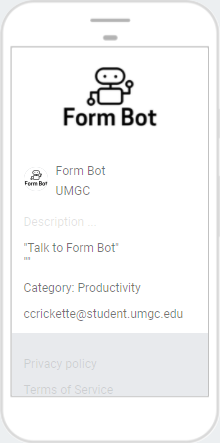


Figure 3 Initial Dialog Window

### Basic Functionality

When the Form Bot application is activated, this screen will show the voice activated input that user evokes.

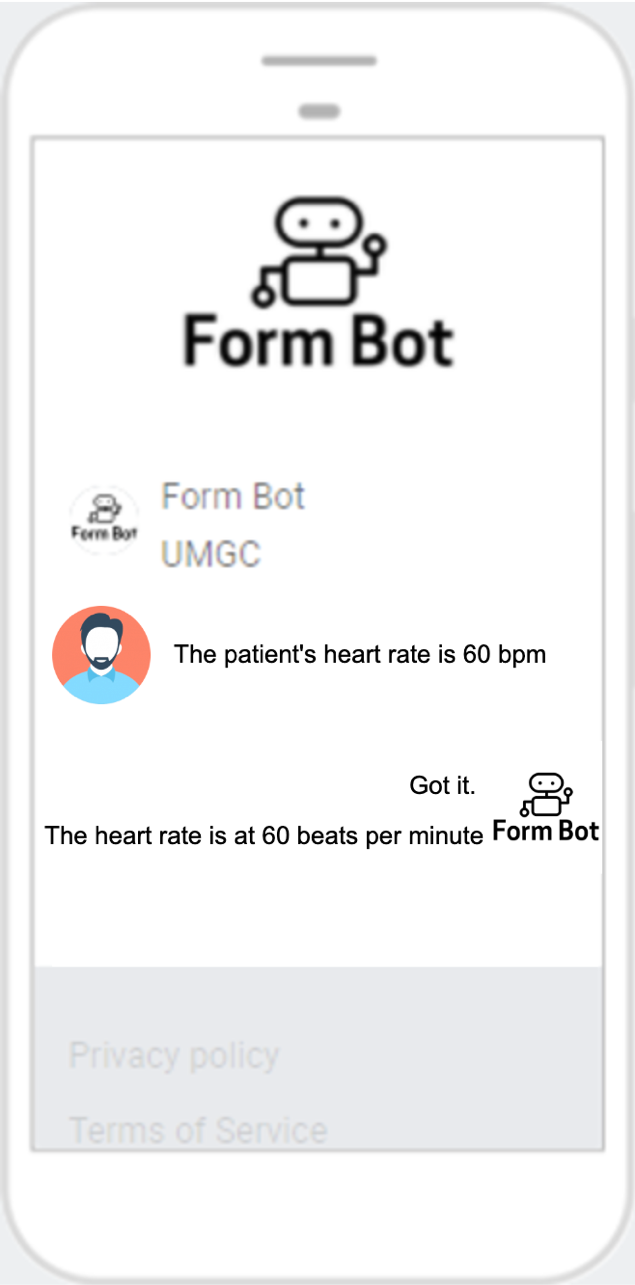


Figure 4 Formscriber AI Functionality

### Web Based Form Management

There shall be a web UI where the user will request the creation of forms, as well as editing and deleting of these forms.

[PROTOTYPE PENDING]

1. Requirement Matrix

This requirements matrix lists all the functional requirements needed for the Form Bot Dialogflow Portion of the Form Bot application.

Table 5 Requirement Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | System Design Component |
| REQ 1.1 | Isolate primary user’s voice for use within the service. | Discern primary user’s voice as the input to the Dialogflow service. | Dialogflow Chatbot Agent |
| REQ 1.2 | Dialogflow service is integrated with the mobile application**.** | DialogFlow engine can be used and accessed by the mobile application. | Dialogflow Chatbot Agent |
| REQ 2.1 | Receive report identifier (id or name) from user. | Identify report identifier from user to receive report metadata. This could be an id or unique name. | Webhook, Application Layer |
| REQ 2.2 | Get report fields from database. | Retrieve report fields from database using report identifier. | Webhook, Application Layer |
| REQ 2.3 | Process report contents and fields. | Process report fields and content into internal data structure. | Webhook, Application Layer |
| REQ 2.4 | Generate dynamic intents to capture report fields. | Use retrieved form fields to generate dynamic intents in Dialogflow. | Webhook, Application Layer |
| REQ 2.5 | Identify report contents extracted by the Dialogflow service. | Identify report contents that are recognized from the user’s voice and dynamic intents using Dialogflow. | Webhook, Application Layer |
| REQ 3.1 | Store extracted report contents into a database. | DialogFlow engine can be used and accessed by the mobile application. | Database |
| REQ 3.2 | Initiate connection with database | Database connection is initialized and any exceptions are handled and presented to the user. | Database |
| REQ 4.1 | Create documentation for more customized chat agents. | Create programmer guide to develop and customize intents for different service professionals. | Dialogflow Chatbot Agent |