Technical Design Document

Milestone 2

Version 1.0

SWEN 670 Software Engineering Project

February 23, 2021

Mobile Team

Presented by: Sylvia Lopez-Willis, Bertina Lee, Brian Malott, Karim ArMansour, Komi Noukafou, Sompon Boontho, Joselito Ocampo, Ermias Seyoum, Arnaud Tako, Alex Dattilio

Technical Design Document Approvals

|  |  |  |
| --- | --- | --- |
| Name | Signature | Date |
| Approved by:  Dr. Mir Assadullah |  |  |
| Approved by:  “Stakeholder” |  |  |
| Approved by:  “Project Manager” | Sylvia Lopez-Willis | 3/16/2021 |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version Number | Date | Description | Approved By |
| 1.0 | 2/14/2021 | Initial Technical Design Document Release | Sylvia Lopez-Willis |
|  |  |  |  |

Table of Contents

[1 Introduction 6](#_Toc64911211)

[1.1 Purpose 6](#_Toc64911212)

[1.2 Scope 6](#_Toc64911213)

[1.3 Overview 7](#_Toc64911214)

[1.4 Reference Material 7](#_Toc64911215)

[1.5 Definitions and Acronyms 8](#_Toc64911216)

[2 System Overview 8](#_Toc64911217)

[3 System Architecture 8](#_Toc64911218)

[3.1 Architectural Design 8](#_Toc64911219)

[3.2 Decomposition Description 9](#_Toc64911220)

[3.2.1 User Interface 9](#_Toc64911221)

[3.2.2 Business Logic Component 9](#_Toc64911222)

[3.2.3 File System 10](#_Toc64911223)

[3.3 Exception Handling 10](#_Toc64911224)

[3.4 Design Rationale 10](#_Toc64911225)

[3.4.1 User Interface 10](#_Toc64911226)

[3.4.2 Business Logic 10](#_Toc64911227)

[3.4.3 File System 11](#_Toc64911228)

[4 Data Design 11](#_Toc64911229)

[4.1 Data Description 11](#_Toc64911230)

[5 Business Logic Component Design 11](#_Toc64911231)

[6 Human Interface Design 12](#_Toc64911232)

[6.1 Overview of User Interface 12](#_Toc64911233)

[6.2 Sign Up 12](#_Toc64911234)

[6.3 Login 13](#_Toc64911235)

[6.4 Logout 15](#_Toc64911236)

[6.5 Begin Conversation 17](#_Toc64911237)

[6.6 Specify “Patient Registration” to Form Bot 17](#_Toc64911238)

[6.7 Specify “New Prescription” to Form Bot 18](#_Toc64911239)

[6.8 Specify “Medical History” to Form Bot 19](#_Toc64911240)

[6.9 Request Report 20](#_Toc64911241)

[6.10 View Reports 21](#_Toc64911242)

[6.11 View Settings 22](#_Toc64911243)

[6.12 View Help 23](#_Toc64911244)

[7 Requirement Matrix 26](#_Toc64911245)

# 1 Introduction

## 1.1 Purpose

The purpose of this document, known as the Technical Design Document (TDD), is to provide a detailed overview of the system design and its architecture that will be used on the UMGC medical Form Bot application. This document will explain how this application will work and the pieces put together to construct its architecture. The intended audience are the Stakeholders, Project Managers, Software Developers, Test Engineers, and Business Analysts from each sub-team of the overall Form Bot project team.

## 1.2 Scope

The scope of this document is to define the technical aspects of the design of the Mobile portion of the Form Bot project.

The overall scope of this project is to process a conversation between a service professional (the end-user) and their respective client into a formatted printable report document, in a secure manner. The overall project requirements specification calls for integration between a mobile application, a Google DialogFlow natural language agent, and a web-hosted management service. The mobile application acts as a starting point for user interaction, connects to the DialogFlow agent for conversation processing, and connects to the Web service for user authorization and report formatting data. The end result is a formatted report following the layout defined by the customer organization in Google Drive.

In Scope:

The scope of the Mobile application portion includes the following:

1. The user interface (UI).
2. Business Logic Component for UI state management, speech recognition, message handling, and remote repository connections.
3. Bi-directional communication with the DialogFlow agent for conversation sessions.
4. Bi-directional communication with the Google API for getting form template URLs and getting and saving completed reports.
5. Bi-directional communication with a Firebase server for user authorization.
6. Application security.

Out of Scope:

The following is out of the scope of the Mobile application project team:

1. Development of the DialogFlow agent, natural language processing, or Artificial Intelligence models.
2. Development of the Web Service, form layout definitions, or web API.

This document aims to provide a clear understanding of the design decisions made regarding the mobile application. This includes the technology to be used during the development phase, a description of the mobile application architecture, data structure definition, and a detailed human interface design.

## 1.3 Overview

This document will cover the system overview and architecture along with the data, component, and human interface designs. Design decisions for the project will be detailed. Descriptions of the contents of each section of this document are as follows:

**System Overview:** The System Overview section addresses the use and purpose of the Form Scriber application. A general description of the available processes will be provided.

**System Architecture:** The System Architecture section addresses the architectural design, design decisions, exception handling, and the design composition of the application.

**Data Design:** The Data Design section addresses how data will be handled in the application.

**Component Design:** The Component Design section addresses how each screen will be composed throughout the application.

**Human Interface Design:** The Human Interface Design section addresses how the application will interact with the user.

## 1.4 Reference Material

This section documents the references used to help with portions of the TDD. The table below has the first column listing the source's title, and the second corresponding column shows where the source was found.

Table 1 - References

|  |  |
| --- | --- |
| Title | Reference |
| Kick-Off Meeting, SWEN 670, Software Engineering Project, Course Homepage | <https://learn.umgc.edu/d2l/home/545048> |
| Project Plan Mobile Team | Sylvia Lopez-Willis |
| Getting Started with the BLoC Pattern | https://www.raywenderlich.com/4074597-getting-started-with-the-bloc-pattern |
| Flutter for Android developers | https://flutter.dev/docs/get-started/flutter-for/android-devs |
| Flutter for iOS developers | https://flutter.dev/docs/get-started/flutter-for/ios-devs |
| Securing Flutter apps | https://danielllewellyn.medium.com/securing-flutter-apps-690100f720db |
| Flutter Secure Storage | https://pub.dev/packages/flutter\_secure\_storage |
| Dart Cross Platform File Storage | https://pub.dev/packages/file\_picker\_cross |

## 1.5 Definitions, Acronyms, and Abbreviations

This section of the TDD will use terms consistent with our mobile application development and customer-requested features.

Table 2 - Abbreviations, Definitions, Acronyms

|  |  |
| --- | --- |
| Abbreviations, Definitions, Acronyms | Definition |
| TDD | Technical Design Document |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| Form Scriber | Name of the application solution. |
| UI | User Interface |

# 2 System Overview

The Form Bot AI mobile application allows public service professionals to process a conversation between themselves (the end user) and the respective client. The application will give the end user the option to view reports from their Google Drive. The application allows the user to view help documents detailing the process of how to use the software. Also, it enables the user to change settings associated with the mobile application. The application uses a Google DialogFlow natural language agent's modern technology to create a functioning AI mobile application.

# 3 System Architecture

## 3.1 Architectural Design

The mobile application system is designed to be a single executable package hosted locally on an Android or iOS device. As such, software development will be conducted using Flutter and Dart within the Android Studio IDE. This allows for containerization of both Android and iOS classes using their respective plugins.

The mobile application consists of the following components: The User Interface (UI), the Business Logic Component, and the file system. The UI encompasses all interaction originating from input by the user at the point of service and ending with output to the file system or printer. The Business Logic Component handles chat session instantiation, external communication, PDF report generation, and print requests. The file system handles report storage and secure access to previously saved reports.

Figure 1: Architectural Data Flow Diagram

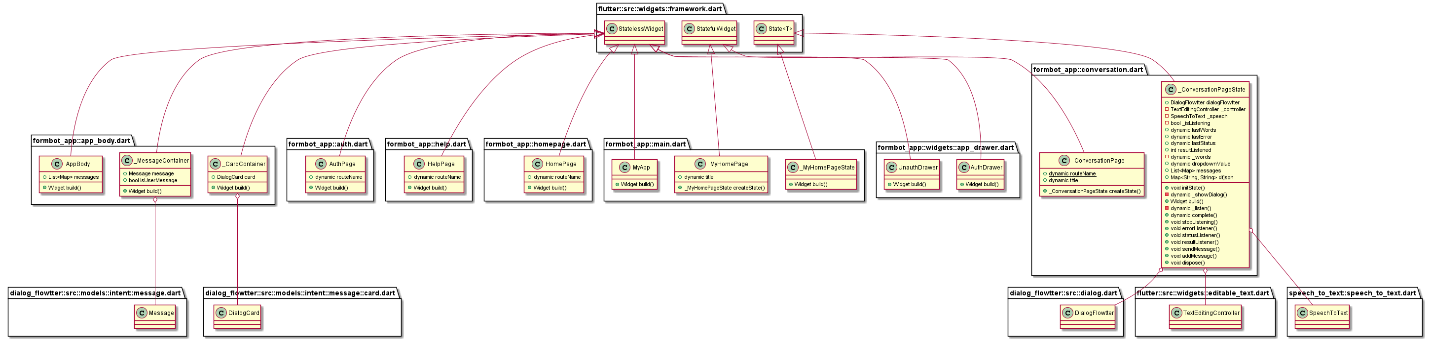
## 3.2 Decomposition Description

### 3.2.1 User Interface

The UI will include input capabilities for the following purposes: user authorization requests, conversation session initialization, conversation audio, settings management, and menu navigation. Users will land on a login screen when the app starts, and after logging in, they will be shown the homepage. From the homepage, users can begin a conversation, manage application settings, view the reports menu, and view the help menu.

### 3.2.2 Business Logic Component

The Business Logic Component is the core engine of the application and controls the logic for handling input data, UI widget states, external communication, and processing output in the form of a report. The UI conversation screen, settings screen, and report view screen all depend on the Business Logic Component for content in some form. Communication between the Mobile app and DialogFlow agent, and requests to the Google APIs are also handled by the Business Logic Component. User authorization will require that users enter a username and password on their screen. This is passed through our Business Logic to the Google API for validation by HTTPS, and after a valid token returns, the UI proceeds to the home page. The business logic will also send HTTPS requests to the Google API for querying completed forms. Messages to the DialogFlow agent are sent by the business logic, and responses are parsed and relayed on screen accordingly.



### 3.2.3 Google APIs

Due to the sensitivity of customer data and privacy concerns being paramount, report viewing is a required application component. Google Drive access will be granted based on authentication tokens managed by Firebase when the user signs in with Google Sign In.

## 3.3 Exception Handling

Exception handling will be implemented on all mobile system layers to properly deal with unexpected parameters, prevent application crashes, unresponsiveness, and various failure modes. The default exception classes within Android and iOS will be used, along with extended exception classes within Flutter and Dart. Where necessary, the exception classes of any third-party imported packages will be implemented as well.

## 3.4 Design Rationale

### 3.4.1 User Interface

The UI will make use of the Material Components library included with Flutter, which is a library of widgets that implement the Material Design guidelines. Material Design is a flexible design system optimized for all platforms including, iOS.

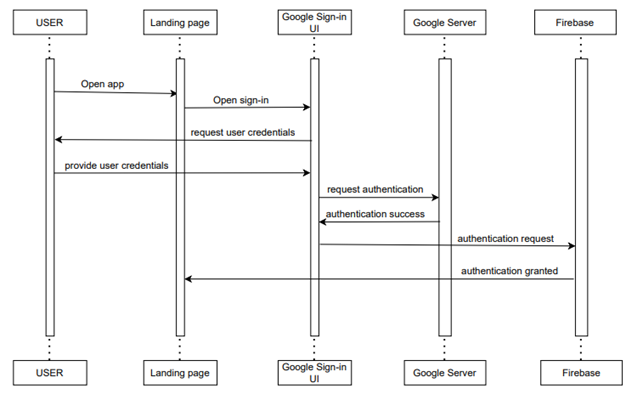
### 3.4.2 Business Logic

The Business Logic Component pattern (BLoC) is a common design pattern coined by Google, which separates the UI views from the core application code. This is most useful for the mobile team because it allows for reusable code across platforms without writing any platform-specific code. Dart was designed to be a platform-independent language but also allow extensibility from external packages. However, the BLOC pattern should be platform-agnostic, meaning no importing platform-specific code logic or dependencies. The solution to this import limitation is by injecting any dependencies.

# 4 Data Design

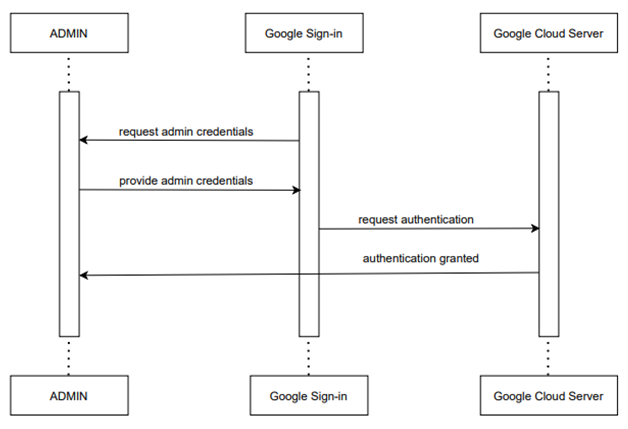
## 4.1 Data Description

The mobile application will not store any raw data locally. The data security concerns applicable to this application are resolved by implementing specific packages provided by Dart and Flutter. The Google Drive API will be used for retrieving templates and storing completed reports. Access to Google Drive will be controlled by authentication tokens managed by the Firebase authentication server. Although the application will not directly save data, there is still a data flow when the user starts the app and try to login. The following data diagram shows how the process will flow from opening the mobile application to the user's authentication.

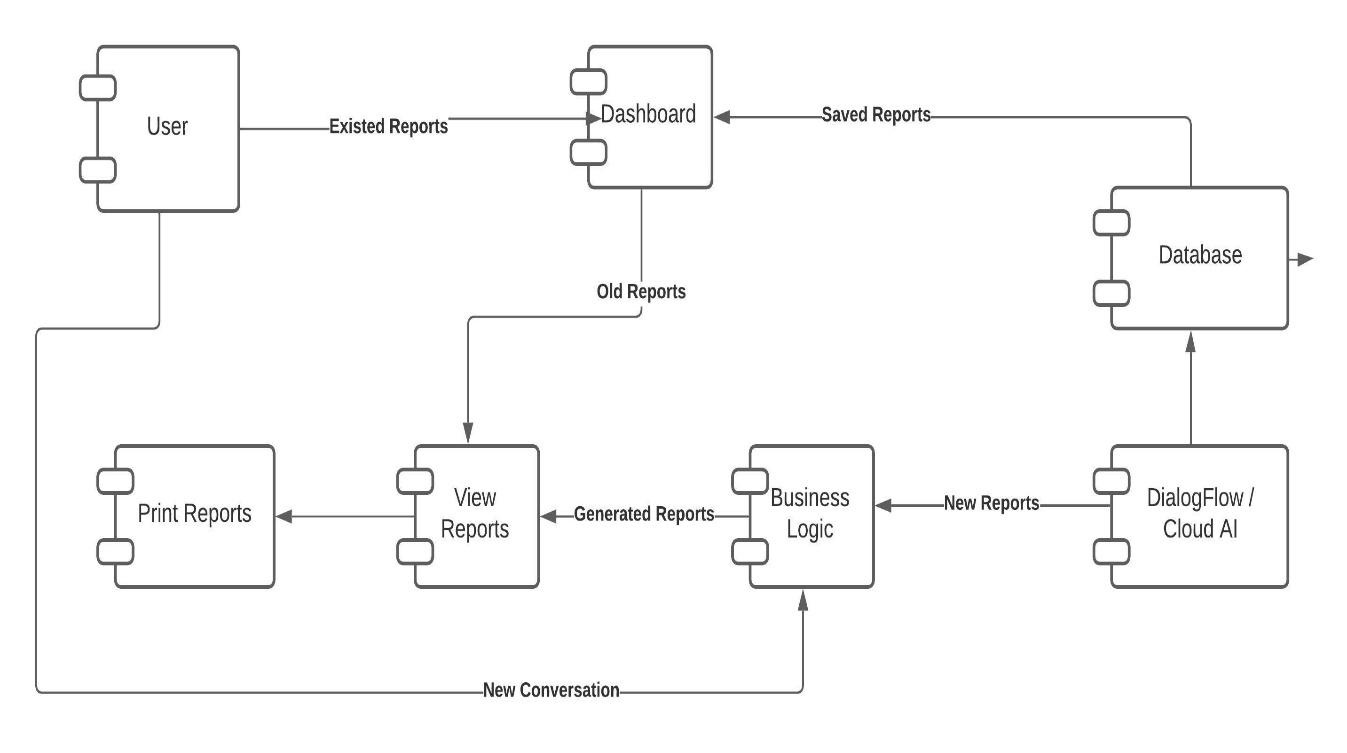


When the user opens the page, he/she will be presented a google sign-in form which the user will fill out and submit. The credentials provided will be sent to the Google server for authentication. On success, the server will authenticate the user. Simultaneously, the credentials will be sent to the Firebase, considered here as Backend as a Service (BaaS) for this application. Upon success, the user will be presented with the homepage, which includes some options and buttons to choose from to begin interacting with the application.

Another data flow that can happen is on the administrators’ side. The administrators will work from the google cloud console. They will have the ability to manage the app functionalities by adding a new user or deleting a user for any given reason. Once the administrators want to do some work related to this application from their google console, the google sign-in form is presented to enter the required information. After submission, the credentials will be conveyed to the Google cloud server for authentication. Once the authentication is successful, the administrator console will be presented to the admins to start their work. The following diagram depicts the administrator side of authentication.



# 5 Component Design



The above component diagram depicts how components are wired together to form more significant components and software systems. They are used to illustrate the structure of arbitrarily complex systems. The user gives the main query, and it converts it into sub-queries and sends it through data dissemination to data aggregators. Results are to be shown to the user by data aggregators. All boxes are components, and arrows indicate dependencies.

Each screen implemented using the BLoC pattern consists of four classes: View, State, BLoC, and Event.

1. The View is responsible for displaying the current state on-screen and delegating input to the BLoC.
2. The State is representative of the current stream of data being presented to the current View.
3. The BLoC is responsible for handling Events, updating the State, and requesting data from local or remote repositories.
4. Events refer to specific actions which may influence State.

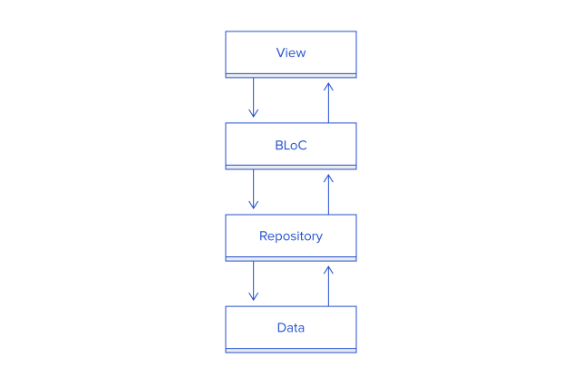
The BLoC architecture has three core benefits.  
It is:

* + Simple
  + Testable
  + Powerful

The BLoC pattern encourages only the use of data Streams and StreamSinks for handling the flow of data between views of the application. User input data will be directed solely to and from the DialogFlow agent and Web Service by secure channels, and PDF reports will be stored in the local secure file system.

The structure of the BLoC will contain the following sub-components:

1. View: Stateful widget code containing draw and layout related design for each screen.
2. Repository: Interfaces for sending HTTPS requests to respective external components and an interface for saving encrypted PDF report files to the secure file system.
3. BLoC: Contains event handlers for user activity, logic and implementation of Repository interface classes to route data queries to external and internal repositories upon a data received event and synchronously or asynchronously update the States and streaming data of affected Views.



**Figure 2:** BLoC pattern component structure.

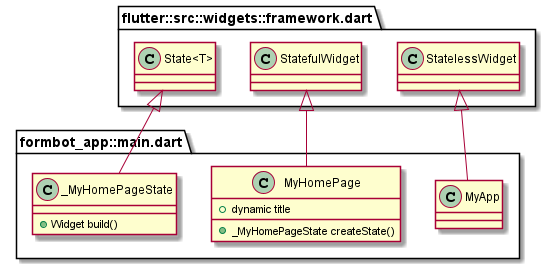
# 6 Human Interface Design

## 6.1 Overview of User Interface

Detailed below are the proposed activities available to the users of the Form Scriber application. The actions below are a guide and could change throughout the development process. The drawings provided are drafts and will change as development progresses.

Each screen will consist of a Widget class, which contains the elements to build on screen, and an optional State class which acts as the logical controller for data to be rebuilt on screen for dynamic StatefulWidgets.

The main class which the application launches into will be a StatefulWidget MyHomePage controlled by the State class \_MyHomePageState.



## 6.2 User Begins Conversation

6.3 Begin Conversation

For the chatbot activity to begin, the user must be logged into the application. The user will select the Begin Conversation option. This option will take the user to the chatbot screen, where communication begins. Once on the chatbot screen, the bot will greet the user and ask for the type of form the user needs. On this screen, the user will have three methods to navigate to the desired form.

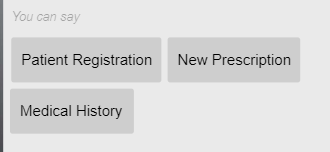
The first option will be to tap the microphone icon (Figure 6.5a). When the microphone icon is pressed, the chatbot will begin capturing audio from the device (Figure 6.5b). A second option will be to tap one of the buttons designated “Patient Registration,” “New Prescription,” and “Medical History.” (Figure 6.5c). A third option will allow the user to type in the form name (Figure 6.5d). A text box will be displayed at the bottom of the screen to receive input. Once the user has completed typing, the user will tap the send icon to send the text to Form Scriber.



*Figure 6.5a - Microphone activated*



*Figure 6.5b - Microphone inactive.*



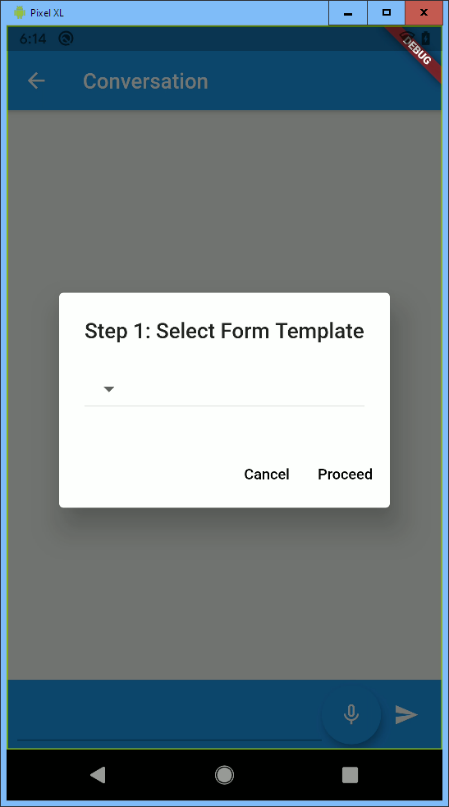
*Figure 6.5c - Form options*



*Figure 6.5d - Form textbox*

## 6.6 Specify a Form Template to Form Scriber

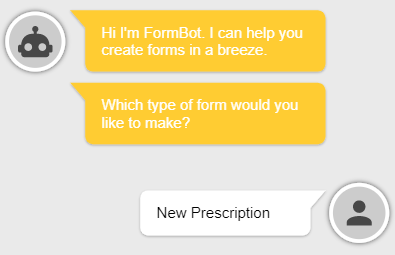
The user will select the template by name from the drop-down menu in the Form Scriber chatbot, then select Proceed. A URL of the template will be sent to the Form Scriber DialogFlow agent. The agent responds with a ready message to indicate the start of the intent collection. The system will start the audio recording after the user taps the microphone button.



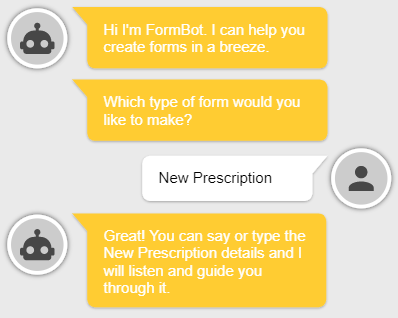
*Figure 6.6 - Form Scriber prompts the user to select from the form template.*

## 6.7 Specify “New Prescription” to Form Scriber

After the user submits the request for the new prescription form (Figure 6.7a), Form Scriber will send the corresponding FormName to DialogFlow. When a response is received from DialogFlow, Form Scriber will prompt the user for the prescription information (Figure 6.7b). At this time, the microphone icon will be activated. The user will have the option to speak or type in the responses.



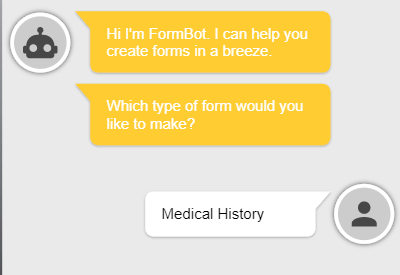
*Figure 6.7a - User specifies “New Prescription” to Form Scriber.*



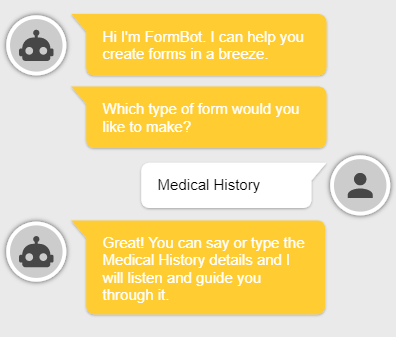
*Figure 6.7b - Form Scriber prompts the user for prescription information.*

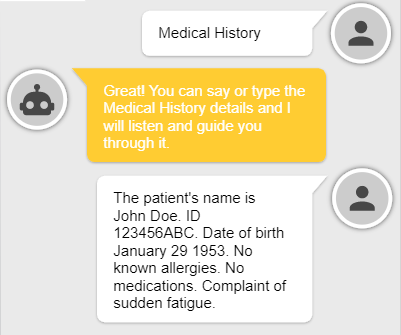
## 6.8 Specify “Medical History” to Form Scriber

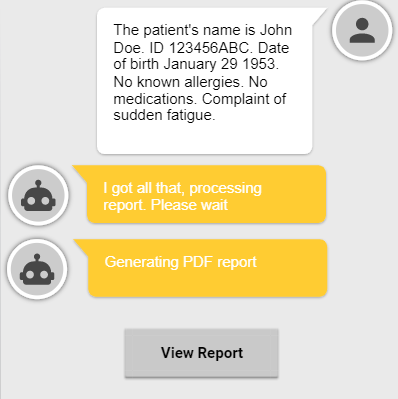
After the user submits the request for the medical history form, Form Scriber will send the corresponding FormName to DialogFlow. When a response is received from DialogFlowerb, Form Scriber will prompt the user for the information to update the medical history (Figure 6.8). At this time, the microphone icon will be activated. The user will have the option to speak or type in the responses.



*Figure 6.8 - Form Scriber prompts the user for information to update the medical history.*

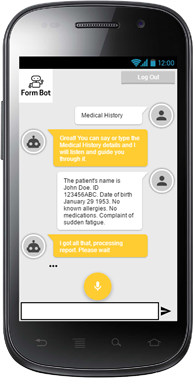






## 6.9 Request Report

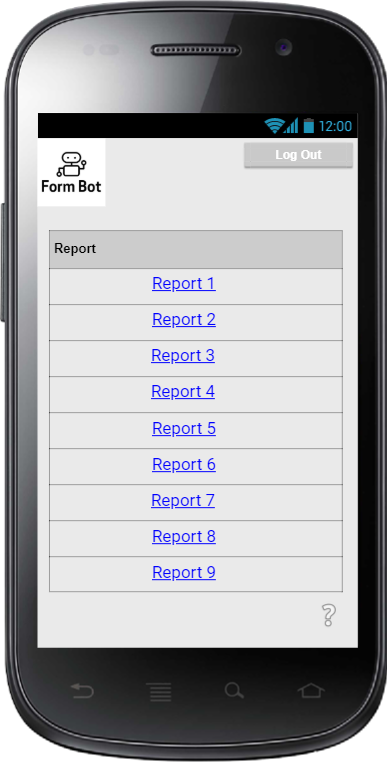
Once the user completes a conversation with the chatbot, the internal service will request a report from the web service API (Figure 6.9). The web API will respond with a JSON string of the report results. This data is then stored for later retrieval.



*Figure 6.9 - The internal service requests a report from the web service API.*

## 6.10 View Reports

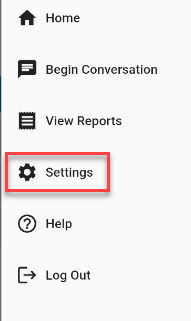
From any screen in the application, the user will have the ability to view existing reports. Form Scriber will generate a list of links to the user's designated Google Drive document repository and display the list of available reports (Figure 6.10).



*Figure 6.10 - Form Scriber displays a list of available reports to view.*

## 6.11 View Settings

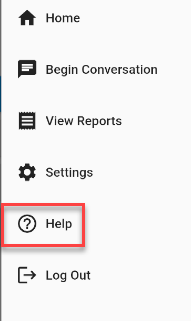
From any screen in the application, the user will have the ability to view the application settings. By selecting the View Settings option, the user will be taken to a screen to view and update the application settings. These settings include selecting dark/light mode and selecting the font family and size for the application.



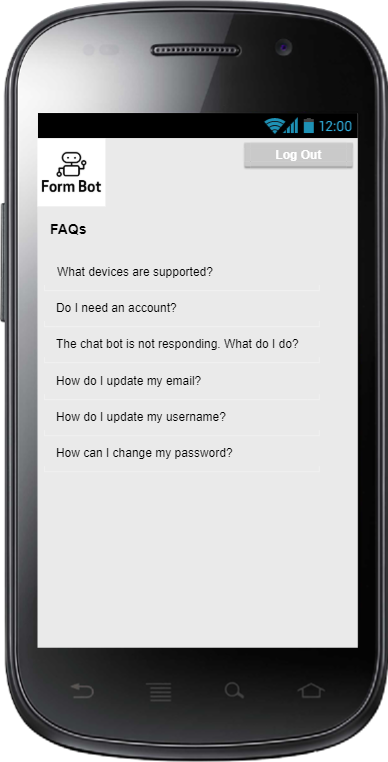
*Figure 6.11 - User is able to select the option to view settings within the application.*

## 6.12 View Help

From any screen in the application, the user will have the ability to view help content regarding the application. The user will select Help from the sliding menu (Figure 6.12a). The user will need to be logged in for access to this feature. The help screen (Figure 6.12b) will consist of Frequently Asked Questions (FAQs). The questions and answers will cover the most commonly asked and anticipated inquiries.



*Figure 6.12a - Help is launched from any screen.*



*Figure 6.12b - The help screen listing the FAQs*

# 7 Requirement Matrix

The requirements traceability matrix for the mobile application of the FormBot project is important to maintain consistency among the various artifacts of the mobile application project. A matrix has been created for the functional requirements of the mobile application. Table 3 shows the number of the requirement, the name of the requirement, and the description of the corresponding name.

Table 3 - Requirement Matrix

|  |  |  |
| --- | --- | --- |
| ID | Name | Description |
| M.1 | Sliding Menu | The user swipes left to right to see the sliding menu options. |
| M.2 | Login | The display and functionality of logging a user into the system. |
| M.3 | Logout | The display and functionality of logging a user out of the system. |
| M.4 | Begin Conversation Session | Navigate to the Form Scriber chatbot activity to begin the conversation session. Connects to Form Scriber to initiate communication. |
| M.5 | Specify a Form Template to Form Scriber | Sends form URL of the selected template to Form Scriber. |
| M.6 | Specify “New Prescription” to Form Bot | The user will trigger a new conversation in the chatbot. Sends URL for “New Prescription” to Form Scriber. |
| M.7 | Specify “Medical History” to Form Bot | The user will trigger a new conversion in the chatbot. Sends URL for “Patient History” to Form Scriber. |
| M.8 | Request Report | Sends form URL of the selected template to Form Scriber. |
| M.9 | View Reports | The user will request to view the report. |
| M.10 | View Settings | The user will request to view settings in the mobile application menu. |
| M.11 | View Help | The user will request to view the Help menu. |