Technical Design Document

Harkify Application

Version 1.0.0

Team Bravo

SWEN 670

Revision History

Author	**Date**	**Reason For Changes**	**Version**
Team Bravo	06/07/2021	Initial version	1.0

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to outline the architecture and design of the Harkify application designed to assist in memory loss. The document will go into the details of the application, specific design decisions and the services that are integrated into the system.

1.2 Scope

The scope of the application is to be an assistant for memory loss. During the minutia of everyday conversation, the application will record and take notes on conversations that the user is involved in. The user will trigger the application using customizable key words and phrases as wake commands to begin the talk to text functionalities. The text will be saved into notes and then be made available for viewing, searching, editing and saving.

The talk-to-text and voice recognition services are supported through external services while the Harkify application is responsible for the storing, searching, and updating of the notes. The UI will allow the customization of settings and wake commands to give the user more freedom to personalize the application. This document will detail the scope of the Harkify application going in depth of the interactions between functionalities.

1.3 Overview

The TDD is broken down into 4 sections that explore every facet of the design. The sections are as follows:

- **System Overview:** This section is a high-level breakdown of the Harkify system. It will provide an overview of the functionalities, service decisions, and context of UI decisions based on requirements.
- **System Architecture:** This section will provide a breakdown of the interactions between services and the design flow that the system follows.
- **Component Design:** This section will discuss the internal services that are required in order to complete the application requirements.
- **UI Design:** This section will provide a walkthrough of the views that the user will encounter using the Harkify application and go in-depth on functionalities and the decisions for those functionalities.

1.4 Reference Material

- Pico Voice services - https://picovoice.ai/
- Azure Speech Recognition - https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speaker-recognition-overview
- Rhino with Pico Voice - https://picovoice.ai/platform/rhino/
- Diagflow TDD- https://teams.microsoft.com/l/channel/19%3A029001df96f442aa80243a05e47227f5%40thread.tacv2/tab%3A%3A376f6312-a3d6-495f-a184-d8f14677a9be?groupId=00b21229-8fbf-4134-a61e-ecdee1866a55&tenantId=d7a68d3a-81dd-434d-bc2b-786d33959b55
- Flutter - https://flutter.dev/docs
- Mark, (2020). Codrus. A Comprehensive Guide to Mobile App Architecture. Retrieved from https://www.coderus.com/a-comprehensive-guide-to-mobile-app-architecture/

1.5 Definitions and Acronyms

TDD - Technical Design Document

UI - User Interface

API - Application Programming Interface

Pico - Pico Voice

2. System Overview

Harkify is a note taking software that integrates talk-to-text and voice recognition software in order to record the user during user specified conversations. The application provides customizable wake words that allow the user to specify when they would like to record their voice. After recording the user can choose to save the note. Due to the likelihood of sensitive information being provided to the application, the notes are encrypted before being stored to local memory. The text in the notes can then be searched by keywords or phrases provided by the user through voice or text. For increased security and performance, the notes are saved for a default seven days however this is also customizable for user preference. The user can prolong the time all notes are saved or can add a favorites tag to specific notes to keep them from being deleted unless the flag is removed. To be able to make sure all points of processing are tested, each service and UI component are unit tested with end-to-end testing available through selenium.

The Harkify architecture can be broken down even further into other components and will be explored further in the following sections. Those components are:

1. Pico Voice/ Rhino/ Azure
2. Note Storage
3. Backend Services
4. UI layer

3. System Architecture

3.1 Architectural Design

The system is comprised of 4 main components: Voice Recognition/talk to text services, Note Storage, Backend setup services and the front-end layer. The Voice Recognition and talk to text are two separate services as both services specialize in each service. The User will interact with the UI, which will send back to any backend service to process the request. Voice notes will be validated and analyzed before saving and storing in local storage.

3.2 Decomposition Description

Each component plays a major role in the overall process of the system. The in-depth diagram of integration is provided in the following sections. The following is the in-depth description of each component.

3.2.1 Azure

Azure Speech Recognition is the leading technology for identifying a user's voice in conversation. By providing a user's voice, through Azure's API, Azure creates a user profile for the specific voice the application is trying to verify. When the audio is submitted, the service will reference the profile to verify the voice is from the user. Azure will send a result object with a result score from 0-1.0 and a result confirmation of accepted of denied. For security, the system saves the voice under an id, as opposed to the name of the user, and requires that Harkify save the id for the user.

3.2.2 Pico/Rhino

Once the voice is confirmed, Pico will handle the voice commands and the talk to text portion the user's notes. Pico and Rhino are a library system that is integrated into the project so that processing can occur quicker than compared to sending everything through an API. While recording a note through a voice will be limited due to the API requirements of Azure, other commands such as search/look up can be done through voice even while offline. Rhino is the tool that will dissect what the user says and pull out the intent of the message. This will help identify important information in each note that could be used later for other tasks like setting up a calendar reminder.

3.2.3 Backend Services

The backend services are not composed of created REST API's but the processing that occurs for every command and the functions that need to occur afterwards. Harkify's backend is written in flutter like the UI, but it adds another layer of security between user and services as it processes and directs the information to the correct channels. Other functionalities include the retrieval of notes, encryption of information and getting the information of notes once saved to a file.

3.2.4 Internal Storage

The way the user will save their notes is through the internal storage of the phone. To accomplish a secure storage the system will encrypt all information before adding it to files but once the file is encrypted, Harkify will store the information in internal storage. this will allow the user to view, save and edit notes that they have put into the system. This can also help with customization of notes and allowing the user to return to notes after the application has been closed.

3.3 Error Handling

All Error handling will be done through UI messaging and Backend processing. If a service is unavailable or the user giving the instructions is not the user on file, the UI will alert the user to try again or inform the user that the services are offline. Services offline only affect the talk to text recording features; however, all other features will still be available if the application is offline because it would all be internal.

3.3 Design Rationale

This section will provide the rationale for every component and service as well as the requirements that are fulfilled by each available component.

3.3.1 Azure

Azure fulfills requirements to maximize security on the user as well as provide a method that only registers the user's voice. The User's voice is only recorded during the stage of creating a profile and user information is never saved. Azure is also a scalable service that provides a free sandbox feature for testing its service but when wanting to scale up begins to charge by audio hour and by every 1000 transactions. It also has complete documentation and videos on how to integrate with your application making it the ideal voice recognition software.

3.3.2 Pico/Rhino

Pico and Rhino fulfills the requirement of talk to text but also adds added functionality to begin to learn and understand more commands from the user. As stated before, this also allows users the ability to use other functions of the application while offline. This is another free service that has a guide for integration. It is also scalable as it becomes a library on the user's device. By splitting the voice recognition and the talk to text, we have allowed users the freedom to use some of the talking features offline.

3.3.3 Backend Services/UI

The Backend services support in the requirements that are fulfilled in the UI. The UI fulfills the requirements that want the user to be able to use the application with ease. Further sections will breakdown the UI components but overall, everything strives to fulfill as many of the requirements as possible.

3.3.4 Internal Storage

Using the internal storage over an online database allows for offline access and guarantees the user more security as all information is only on their personal device. Internal storage fulfills the requirements of increased security for the user especially due to PII being part of the notes.

4.0 Data Design

This section outlines the principal data structures used by Harkify to store and transport information for the application. Each data structure is described in detail, including all properties and acceptable data formats.

4.1 Data Description

4.1.1 Text Notes

Harkify saves all text notes derived from user speech in separate text files. Each file name starts with "textnote_" followed by the date and time when the note was recorded so that users may identify text notes outside of the application, if necessary. Data within each file is maintained in an XML format. All text note operations are handled by the TextNoteService class. The following XML data represents a sample format for a text note:
<?xml version="1.0"?>
<text-note>
<file-name></<file-name>

```
<when-recorded></when-recorded>
<is-favorite></is-favorite>
<text></text>
</text-note>
```

4.1.2 Application Settings

Harkify stores all application settings in a text file called "settings.txt". As with text notes, the data within the file is maintained in an XML format. All operations against application settings are handled by the SettingService class. The following XML data represents a sample format for the settings text file:

```
<?xml version="1.0"?>
<settings>
<note-purge-days></note-purge-days>
<end-save-seconds></end-save-seconds>
<save-triggers>
<save-trigger-phrase></save-trigger-phrase>
<save-trigger-phrase></save-trigger-phrase>
<save-trigger-phrase></save-trigger-phrase>
</save-triggers>
<end-triggers>
<end-trigger-phrase></end-trigger-phrase>
<end-trigger-phrase></end-trigger-phrase>
<end-trigger-phrase></end-trigger-phrase>
</end-triggers>
<search-triggers>
<search-trigger-phrase></search-trigger-phrase>
<search-trigger-phrase></search-trigger-phrase>
<search-trigger-phrase></search-trigger-phrase>
</search-triggers>
</settings>
```

4.2 Data Dictionary

Entity	Field	Description	Data Type	Default
Text Note	File Name	File name of the text note saved	String	
Text Note	When Recorded	The date and time when the text note was saved	Datetime	
Text Note	Is Favorite	Whether or not this text note has been favorited by the user so it will not be purged	Boolean	False
Text Note	Text	Text of the user’s speech recorded as a text note	String	

Settings	Note Purge Days	Number of days before an unfavorited text note is purged from the system	Integer	7
Settings	End Save Seconds	Number of seconds of silence before a text note being saved ends as though an end trigger phrase were detected	Integer	5
Settings	Save Triggers	List of trigger phrases to make the system begin saving the user's speech as a text note	List of Objects	
Settings	End Triggers	List of trigger phrases to make the system end saving the user's speech as a text note	List of Objects	
Settings	Search Triggers	List of trigger phrases to make the system search for matching text notes, filtered by the user's speech	List of Objects	
Save Triggers	Save Trigger	A trigger phrase to make the system begin saving the user's speech as a text note	String	
End Triggers	End Trigger	A trigger phrase to make the system end saving the user's speech as a text note	String	
Search Triggers	Search Trigger	A trigger phrase to make the system search for matching text notes, filtered by the user's speech	String	

5. Component Design

Defining mobile application architecture assists in building a successful application and gives both the developers, and all the stakeholders involved, a clear understanding of the overall process, work, and data flow of the application. Walking through the application architecture provides and later ensures the application is both reliable and scalable. In the future, any business requirement changes will be more manageable to do by using the initial architectural design as a guide. The diagram below is a general guide to a common mobile application. (Mark, 2020)

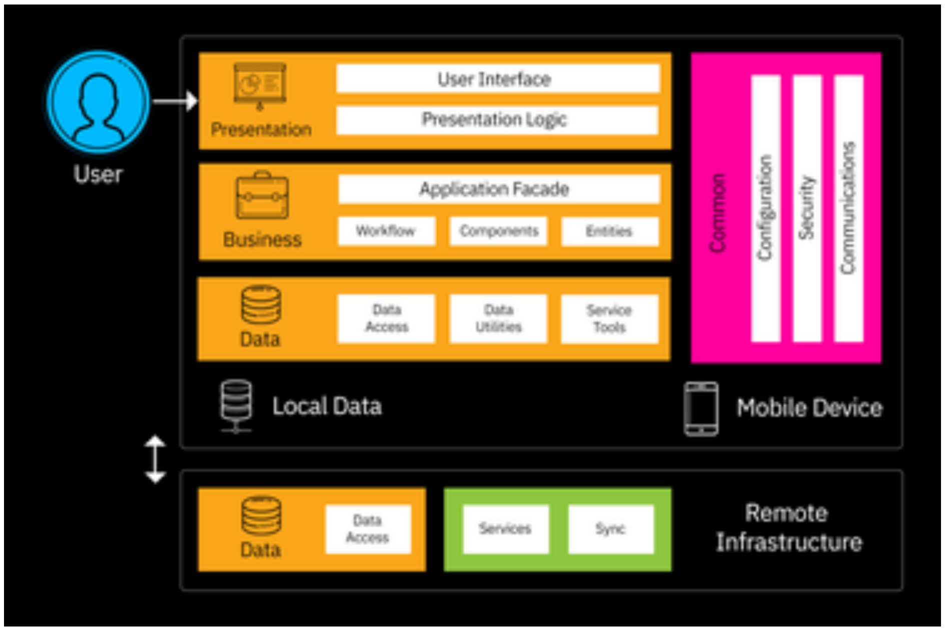
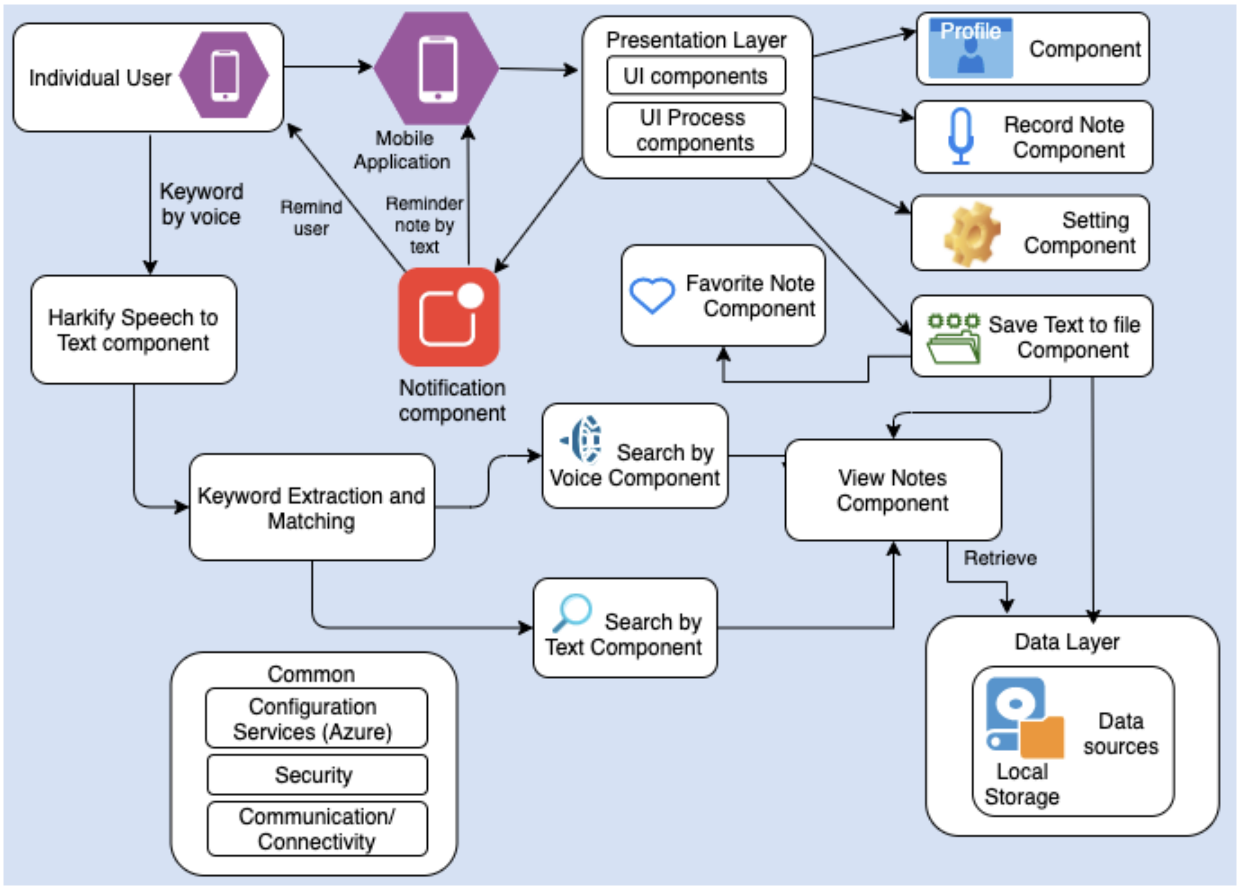


Diagram of a Mobile Application

Important elements and factors that should be considered before building the mobile application architecture design include: the device type, bandwidth scenarios, navigation methods, UI, and push notifications.

- Device Type - the application will run on both android and IOS devices. A certain amount of memory will be needed for storing data for a customizable time period on the local device. The flutter platform used for the Harkify application helps maintain quality across all operating systems.
- Bandwidth Scenarios - the Harkify application is designed to be available for use with an internet connection for initial start-up. After initialization of the user profile, the application will then be able to support offline users.
- Navigation Methods - the Harikify application is designed using the various navigation options available by the flutter framework which include Direct Navigation with MaterialPageRoute, Static Navigation with Route Map, and Dynamic Navigation with Generated Routes. Some of the navigation types to include are hamburger menu, gesture-based navigation, floating action button, and more available on flutter.
- User Interface (UI) - Harkify is the center point of interaction for the user and functions as the basic presentation layer. The UI is further described below in the human interface section of this technical design document. From the customer's viewpoint, the UI is the key to their satisfaction. Mock views are available in the UI section below.
- Real-Time Update vs Push Notification - the application will constantly engage the user using push notification to utilize the application. If there is no interaction with the system functionality for the past twelve hours, then the application will send a push notification that can be enabled and disabled by the user as needed.

The front end of the Harkify application is built using the flutter framework to provide users with an easy-to-navigate interface. With Flutter, various elements are divided up into components. The figure below displays the different layers of the Harkify mobile application architecture starting from the presentation layer, business layer, service layer, and data access layer.

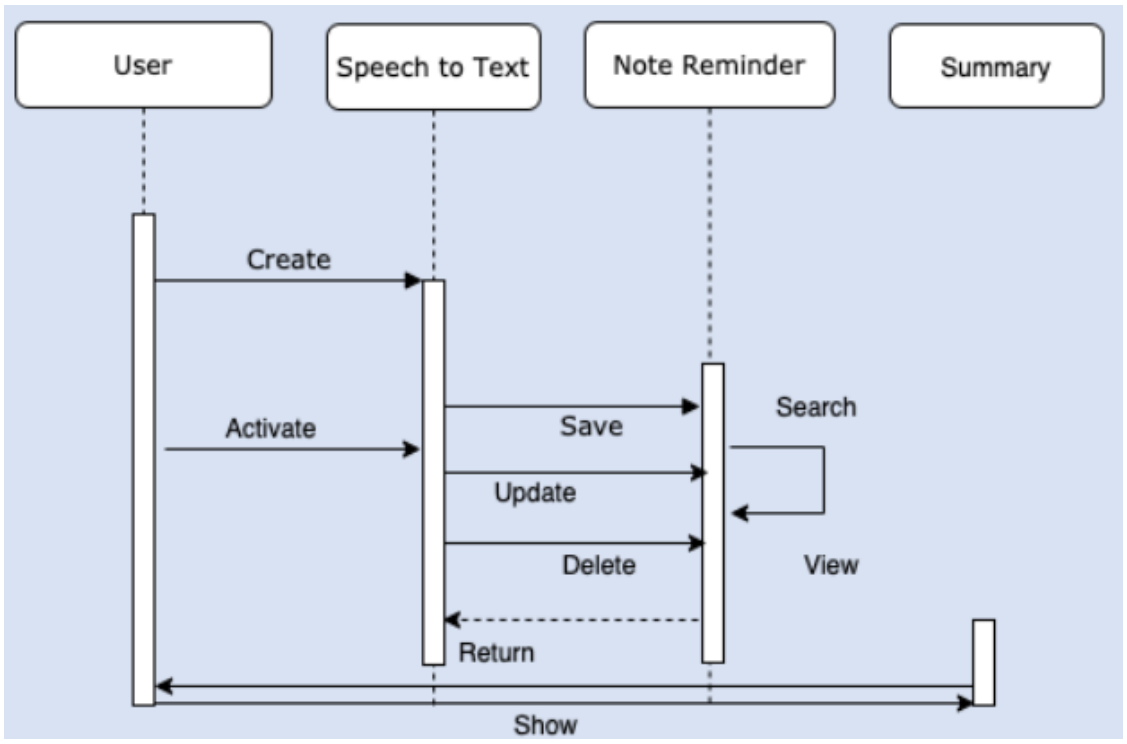


Elements of Harkify Application Architecture

The mobile application design above helps address separate components of the Harkify application into distinct sections to address each section with separate concerns. Each module is responsible for only specific features/functionality. The specific functionality should not be repeated by other components during development to reduce repetition of the software application pattern which in turn helps avoid defects. It should instead be replaced with available abstractions or data normalization to avoid this redundancy. The application is designed with clarity and simplicity allowing for possible design evolution over time. Factors taken into account during the overall architectural design of the Harkify application include efficiency, flexibility, scalability, extensibility, understandability, and testability. All the stakeholders involved are responsible to make sure this checklist is satisfied before the deployment of the application.

As shown in the *Sequence diagram for Harkify Application* below:

- The user creates a new profile using the welcome page.
- The user activates the mobile device using keyword trigger or tap action to start recording.
- The speech from the specific user is converted to text and later saved in a text note format into the local storage.
- The system saves the converted speech to text in a file where it may also be updated and deleted by the user from the existing recorded entries.
- The information is returned to the reminder note process by the local storage.
- The Summary process is invoked to view the summarized note information of the reminders as added by the user.
- The summarized information is returned to the user in return for this invocation.



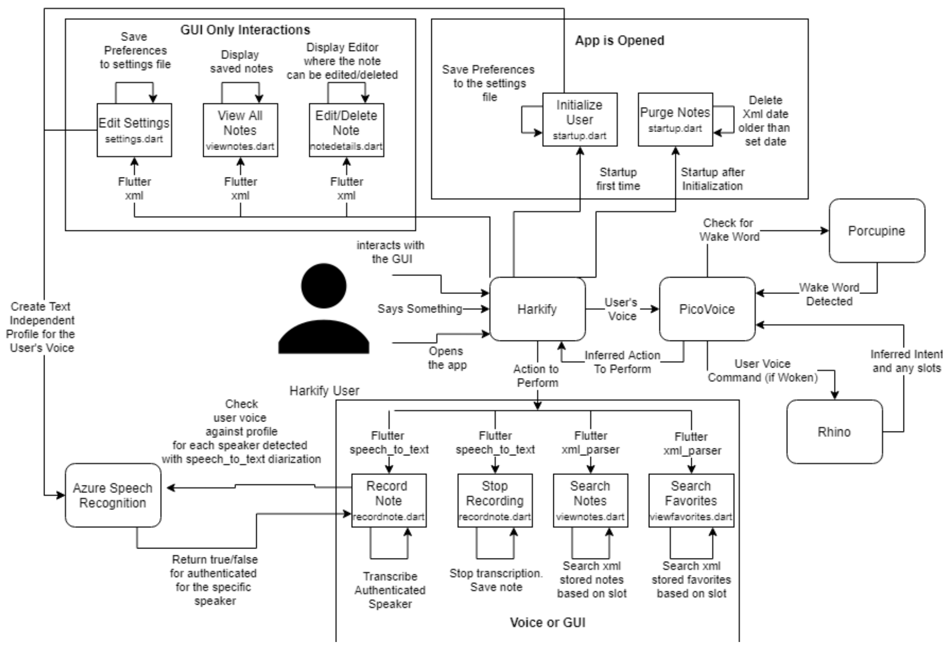
Sequence diagram for Harkify Application

6.0 Human Interface Design

6.1 Overview of User Interface

In this section all the graphical user interface components and user interactions with the Harkify application will be outlined. Many of the proposed graphical components are visual mock-ups and will change during application development. The Graphical User Interface and user interactions described in this section are merely a guideline and may change during the course of development.

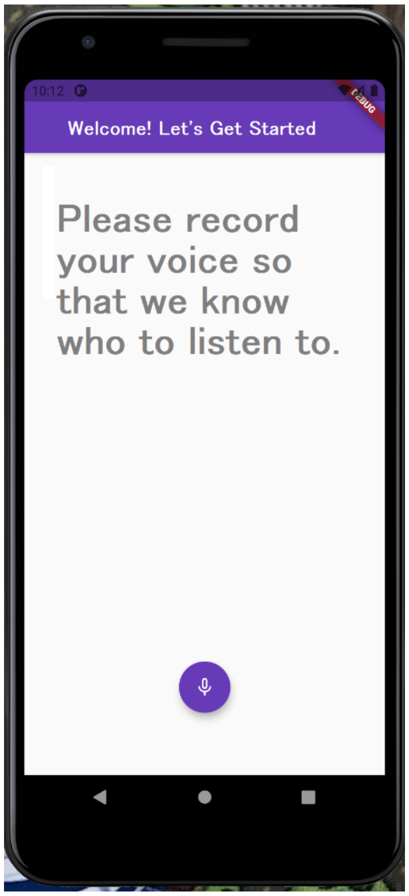
Every screen of the Harkify application is encapsulated in its own *.dart* file. Each *.dart* file containing classes for application screens extends the Flutter *StatefulWidget*. These widgets contain the visual elements which are displayed to the user for each given screen, along with the backend logic to be performed while the user is interacting with the screen. A high-level diagram outlining the screens with their corresponding *.dart* files and user interactions can be seen below in *Figure 1*.



6.2 User Initializes Application

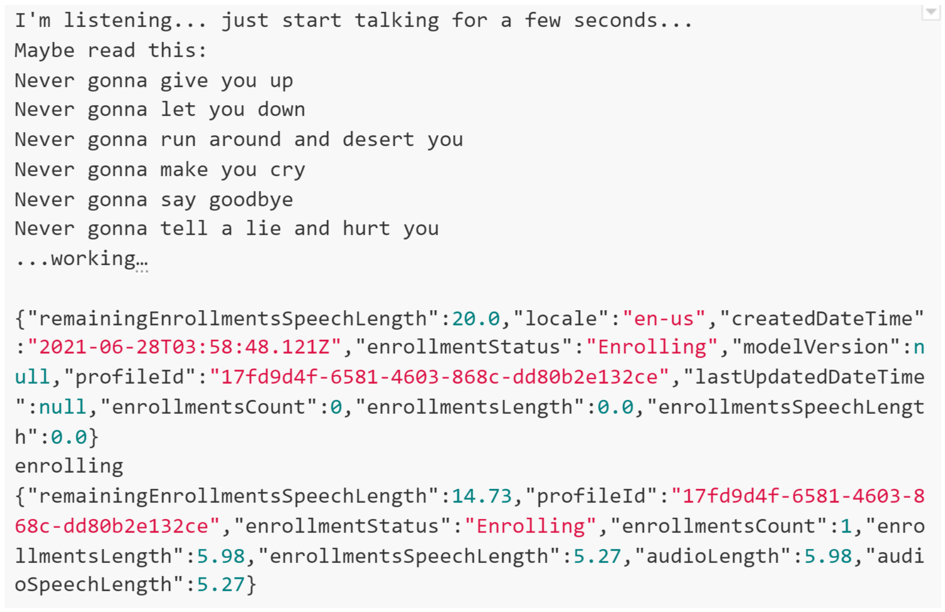
To launch the app, the user will click on the application logo tile from their mobile device, pictured below in *Figure 2*.

In order for the application to work, the user must first create a voice profile with *Azure Speech Recognition*, after which we will verify that we can verify the user's voice based on the profile they have created. The screen shown in *Figure 3* will be displayed to the user, and clicking the microphone button will start the recording and user profile creation process.



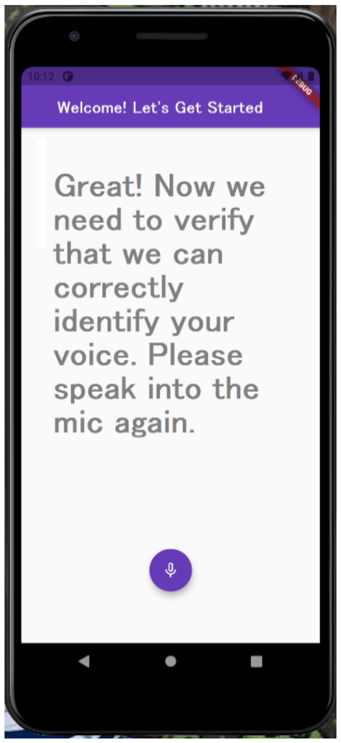
Please record your voice so that we know who to listen to.

After submitting the voice information to the API via HTTPS request, a return similar to what is displayed in *Figure 4* will be returned.



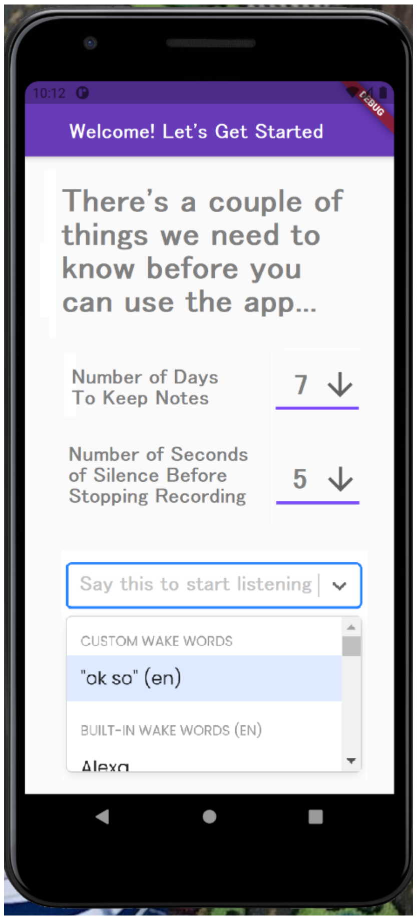
```
I'm listening... just start talking for a few seconds...
Maybe read this:
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
Never gonna tell a lie and hurt you
...working…

{"remainingEnrollmentsSpeechLength":20.0,"locale":"en-us","createdDateTime"
:"2021-06-28T03:58:48.121Z","enrollmentStatus":"Enrolling","modelVersion":n
ull,"profileId":"17fd9d4f-6581-4603-868c-dd80b2e132ce","lastUpdatedDateTime
":null,"enrollmentsCount":0,"enrollmentsLength":0.0,"enrollmentsSpeechLengt
h":0.0}
enrolling
{"remainingEnrollmentsSpeechLength":14.73,"profileId":"17fd9d4f-6581-4603-8
68c-dd80b2e132ce","enrollmentStatus":"Enrolling","enrollmentsCount":1,"enro
llmentsLength":5.98,"enrollmentsSpeechLength":5.27,"audioLength":5.98,"audi
oSpeechLength":5.27}
```

Once the user profile has been created, we will need to verify that the user can be successfully identified. The screen shown in *Figure 5* will be shown to the user. Again, the microphone button will initiate recording and submit the details for verification. All the user profile information is stored within Azure Speech Service for the application, and is not localized to the device.



Great! Now we need to verify that we can correctly identify your voice. Please speak into the mic again.

After the voice profile has been successfully set up, the user will be shown the app configuration settings and given the option to change them from the default values before starting to use the app. These values are: The number of days that transcribed notes will be saved, the number of seconds of silence before a transcription is automatically stopped, and the wake word that the user will use to tell the app to start listening to them.



10:12

Welcome! Let's Get Started

There's a couple of things we need to know before you can use the app...

Number of Days To Keep Notes: 7

Number of Seconds of Silence Before Stopping Recording: 5

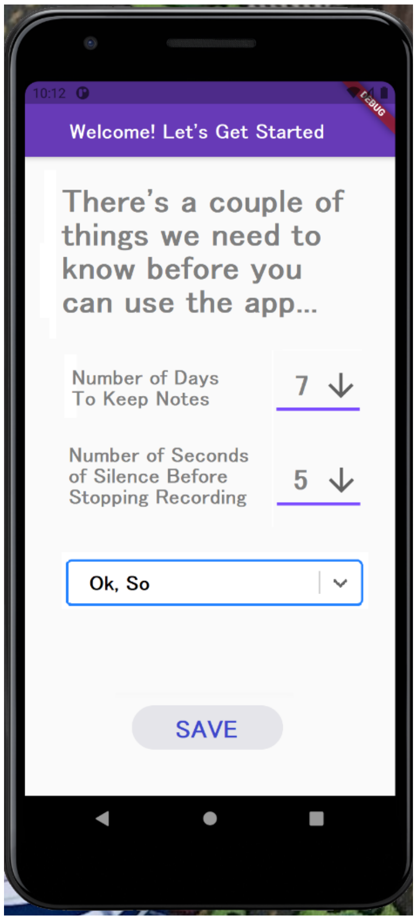
Say this to start listening

CUSTOM WAKE WORDS

"ok so" (en)

BUILT-IN WAKE WORDS (EN)

Alexa



10:12

Welcome! Let's Get Started

There's a couple of things we need to know before you can use the app...

Number of Days To Keep Notes: 7

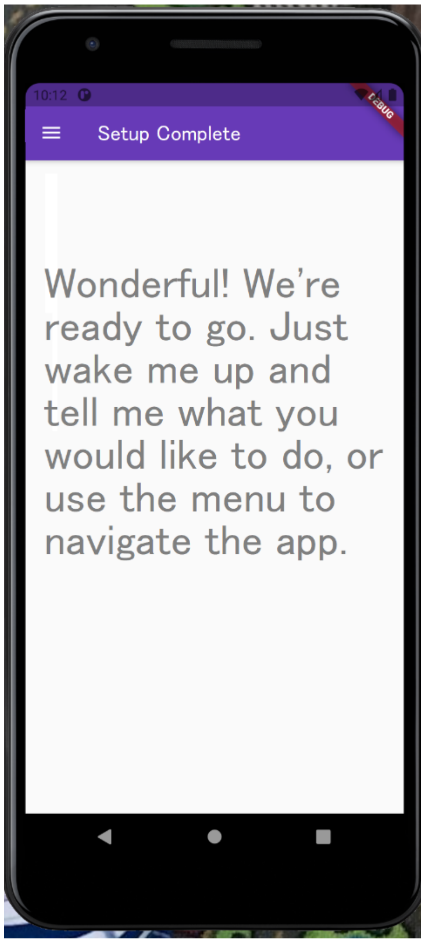
Number of Seconds of Silence Before Stopping Recording: 5

Ok, So

SAVE

The default settings are pre-selected so the user can merely press “save” if they do not want to change anything. These settings can also be changed at any time through the settings page. These settings details will be saved to an xml file and stored locally, to be read every time the app is started.

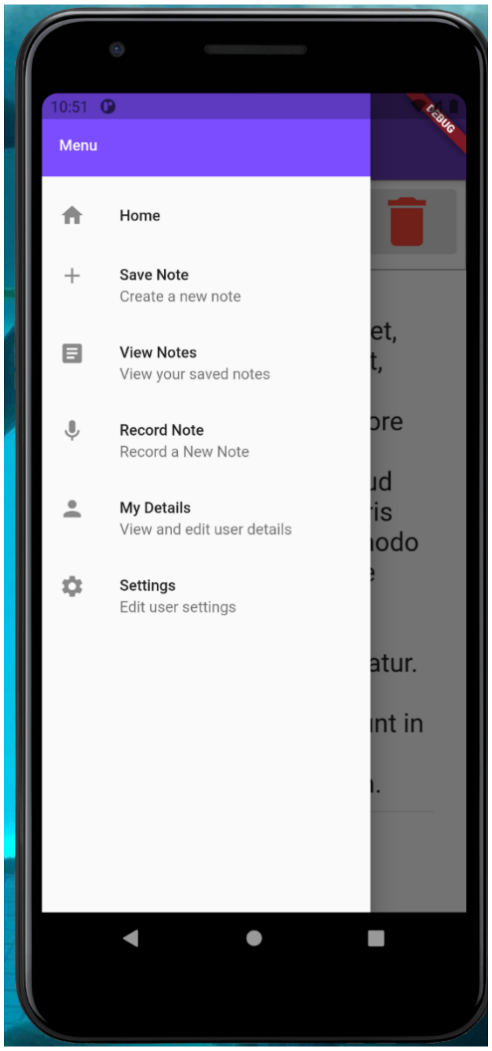
After the settings are saved, the user is shown a confirmation message letting them know that they are ready to use the application. This would also be the screen in which walkthrough videos would be displayed to the user to explain app usage.



Wonderful! We’re ready to go. Just wake me up and tell me what you would like to do, or use the menu to navigate the app.

6.3 User Opens the Navigation Menu

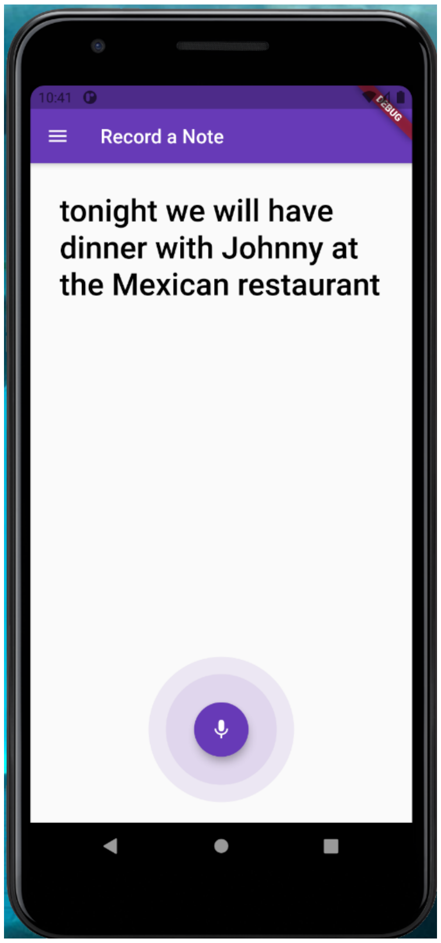
Once the user has successfully initialized all the app settings, they can navigate the app through the application menu. This is opened by clicking the menu icon in the top right hand corner of the application, and the widget code for the menu is found in *basemenudrawer.dart*. The menu control is a *Drawer* containing a *DrawerHeader* and the menu options are *ListTile* controls utilizing the *onTap()* event to navigate to the various application routes.



6.4 User Records a Note

When the user wishes to record a note, they have two ways to start the transcription. They can either navigate to the “Record Note” option within the menu and press the microphone icon, or they can say the application wake word (“Ok, so” by default) and then say a phrase that will be interpreted by the NLU tool *Rhino* to infer that the user wishes to begin transcribing. An example of this would be “Ok, so *[the application begins listening]* I need to remember…” or “Ok, so *[the application begins listening]* What you are saying is…” etc. The *Rhino* profile for the application will be trained with many possibilities for inferences so that the user can discreetly tell the application what they would like, and the application is able to correctly comply without needing the user to set and remember triggers.

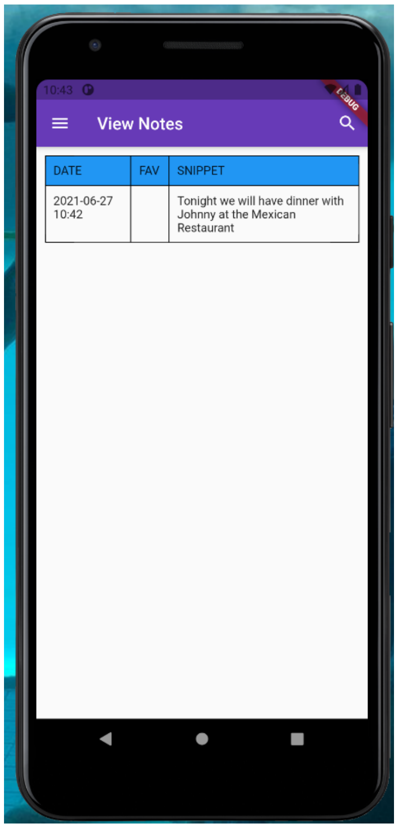
Once it is inferred that the user wishes to start recording their conversation, or the microphone icon is clicked, each speaker identified using Flutter's *speech_to_text* diarization must be authenticated using *Azure Speech Recognition* verify functionality so that the application user's is the only voice that is transcribed. Once the speaker is identified, *speech_to_text* will be utilized to transcribe the user's voice until enough seconds of silence have elapsed, or a specific end trigger is spoken. This note is then automatically saved as an xml note that the application can access and display to the user. A demo of this functionality can be observed here.



tonight we will have dinner with Johnny at the Mexican restaurant

6.5 User Views Recorded Notes

The user can view their recorded notes by selecting “View Notes” from the navigation menu. A date-sorted Flutter *Table* of stored notes will be displayed to the user. Tapping any note record will open that note so that the user can view the full transcription of the note, as well as perform any edit or delete actions.



DATE	FAV	SNIPPET
2021-06-27 10:42		Tonight we will have dinner with Johnny at the Mexican Restaurant

6.6 User Performs a Search

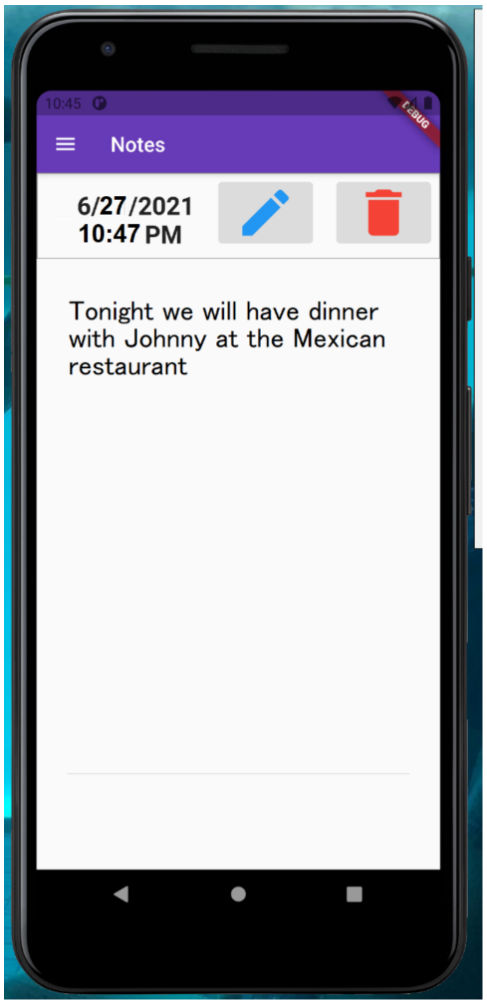
There are two options for the user to perform a search on their notes or Important Details. They can either use the search bar - created using *flutter_search_bar* - in the user interface, accessible after tapping the search icon in the top left hand corner of the screen, or by saying the application wake word and then say a phrase that will be interpreted by the NLU tool *Rhino* to infer that the user wishes to begin searching. An example of this might be for example the user says "Ok, so *[application begins listening*] what is my medicare number?" and rhino would return a JSON object similar to *Figure 12*.

```
{
intent: "lookupFavorite",
slots: {
info: "medicare number"
}
}
```

Once a search has been initiated, *xml_parser* will be utilized to search the stored files for the user's transcribed search phrase.

6.7 User Views a Specific Note

Specific notes can be viewed by tapping the snippet displayed in the "View Notes" page either from the ordered list or search results. The full transcription will then be displayed to the user, along with the date that the note was transcribed, and buttons which permit the user to edit or delete the transcription with the *xml* package.

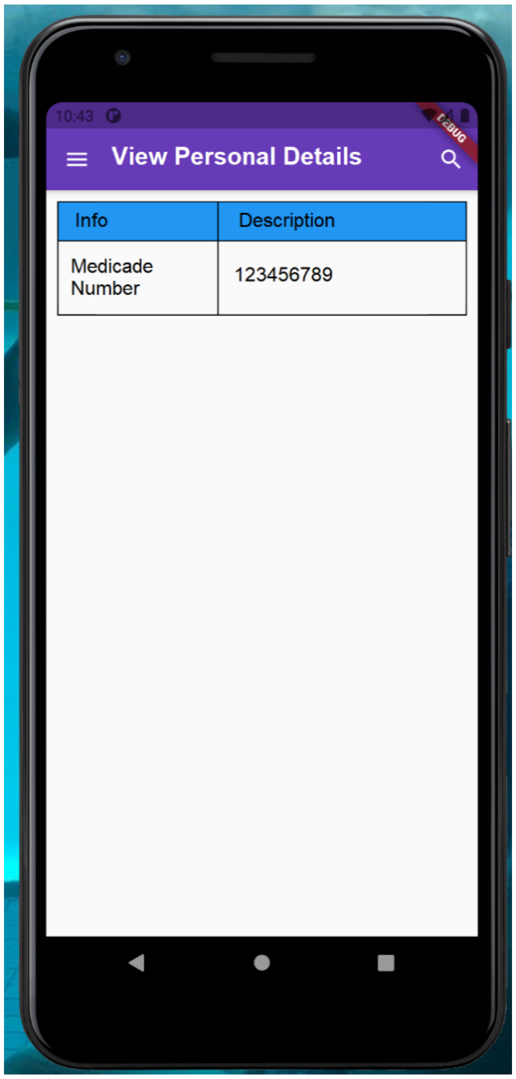


6/27/2021
10:47 PM

Tonight we will have dinner with Johnny at the Mexican restaurant

6.8 User Views Their Important Details

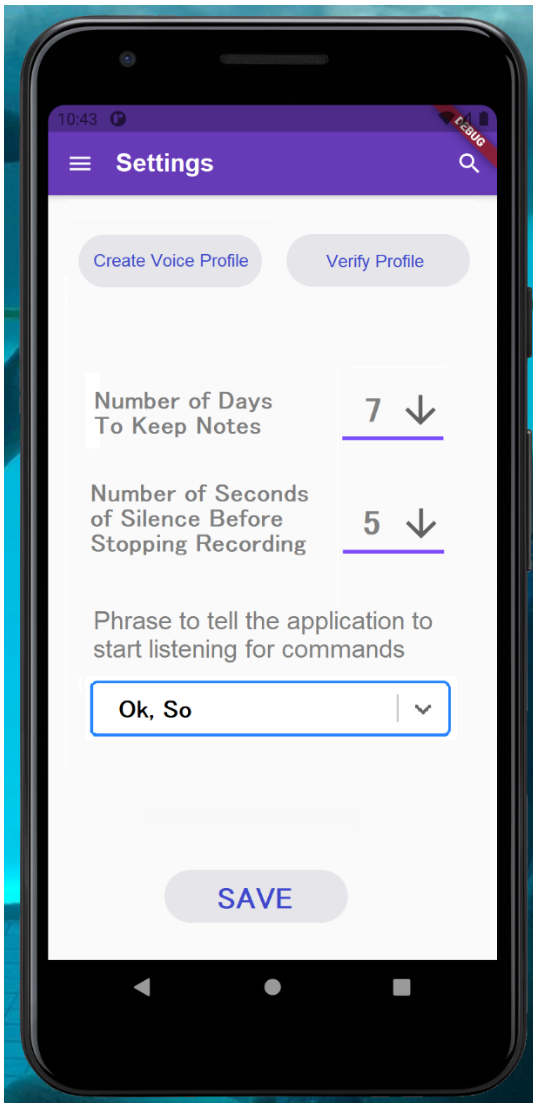
This functionality is identical to the “View Notes” page, with the key difference being that instead of the user viewing transcribed notes, they will be able to view, edit and delete their important details as key-value pairs within a Flutter *Table* control. The key is the name of the setting, such as Social Security Number, Medicade Number, Address etc. and is set as a searchable slot within *Rhino* called “info.”



Info	Description
Medicade Number	123456789

6.9 User Views Their Settings

The user can change any of the settings that were set during app initialization, including setting the voice authorization, through the “Settings” page accessible through the navigation menu. This will allow the user to create a new voice profile in Azure, as well as edit the settings that are stored in the local xml file.



Number of Days To Keep Notes 7

Number of Seconds of Silence Before Stopping Recording 5

Phrase to tell the application to start listening for commands

Ok, So

7.0 Requirements Matrix

Below is a list of requirements for the project created by all teams:

Requirement	Mandatory/Optional
1.) The application shall not save any voice recording.	Mandatory
2.) The application shall allow the user to edit any speech converted to text.	Mandatory
3.) The application shall provide the option to convert the user's speech to text.	Mandatory
4.) The application shall provide an user interface that incorporates the following device features: Bold Text, Display Zoom, the ability to Increase Text Size, and High Contrast Colors. (Optional)	Optional
5.) The application should read the text and play the synthesized speech into the user's earbuds.	Mandatory
6.) The application shall keep the end user persona in mind for UI/UX considerations and offer a flexible environment for the user to customize.	Mandatory
7.) The application shall provide training videos within the app to guide the user regarding its various features and functionalities.	Mandatory
8.) The application shall recognize distinct phrases and sentences that he or she uses while speaking to him or herself or with others.	Mandatory
9.) The application shall learn the phrases that the user wants to use when talking to someone while trying to save important spoken text and also the phrases that the user wants to use speaking to him or herself trying to retrieve the noted information.	Mandatory
10.) The application shall provide a facility to the user. This facility will note several phrases that the user may use in natural conversation to save as well as to retrieve saved information to play as speech.	Mandatory
11.) The application shall ignore everything except what the user speaks.	Mandatory
12.) The application shall provide the means for the user to train the app on its voice.	Mandatory
13.) The application shall bypass asking everyone permission to record.	Mandatory
14.) The application shall provide the ability to save the speech to conversion texts to local device storage.	Mandatory
15.) (Optional) – The Mobile Operating System's Virtual Assistant shall interface directly with the Application by Application Name	Optional
16.) The application shall provide the following means to activate recording: Tap on the app and immediate voice recognition.	Mandatory
17.) The application shall provide the ability to search through the saved speech to text notes via text field and/or voice command (Optional).	Optional
18.) The application shall retrieve all results related to the search command.	Mandatory

19.) The application shall retain speech to text recognition notes for 1 week in duration.	Mandatory
20.) The application shall provide a trigger to end the voice recording.	Mandatory
21.) The Application must work with IOS and Android	Mandatory
22.) The Application must encrypt the data at rest.	Mandatory