Technical Design Document

TEAM: CHARLIE

Presented by: Michael Le, Debashis Jena, Austin Johnson, Prince Antwi Aboagye, Didimus Kimbi, Damion Sevilla

SWEN 670 – SOFTWARE ENGINEERING PROJECT
JUNE 29, 2021
REVISION 2.0

Project name: Mnemosyne, Disability Mobile Application

Date: June 29, 2021

Project Leader: Michael Le

Phase: Design & Engineering and Execution

For approval: Michael Le

Michael le_____ Date: 06/29/2021

For approval: Dr. Mir Mohammed Assadullah

_______ Date: 07/02/2021

Revision History

Version Number	Date	Description	Author(s)	Approved By
1.0	06/29/2021	Initial Technical Design Document	SWEN670 (Summer, 2021) - Team Charlie	PM – Michael Le
2.0	08/06/2021	Updated with latest design changes	Debashis Jena	PM – Michael Le

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to show the design and architecture detail of the Mnemosyne mobile application. The design detail led to the implementation of the application base on the feature functionalities, user interfacing information, and service interaction between application technologies.

1.2 Scope

The overall scope of the Mnemosyne application is to allow service professionals and disabled people (the users) to record specific speech conversations for retrospective purposes. The recorded audio is processed, and data is extracted from the combination of the management Mnemosyne application, user devices, and Google Cloud service. The processed information is then stored into a local storage device database that measures highly encrypt and decryption. The data can save for at least a duration of time as an official record and depend on user desires things to do with the application.

Team Charlie committed to exploring application technologies from setting up the analytical speech recording engine using Google's Cloud service. The overall Mnemosyne application system contains processing and storage of user's voice data will be functional. This document details the internal architecture for the Mnemosyne mobile application service that demonstrated breaking down the level of detail on different speech functionalities and suffice the original requirement from the customer.

1.3 Overview

The TDD has five sections that described the significant dimensions of the Mnemosyne mobile application service architecture. These sections are broken down as specified below:

- **System Overview**: This is the high-level description of the overall Mnemosysne system, its described the functionality, decisions, contexts that in the document implementation.
- **System Architecture**: This section provides a breakdown of the different portions of team Chalie services to ensure the Mnemosyne application fully functional. Its investigation of the level of detail functionalities of the description and demonstrated how it implement to fulfill the given requirement.
- **Data Design**: This section mainly focuses on how the data implements and identifies relational databases to make the program run smoothly.
- **Component Design**: This section focuses on the detail of specific components broadly, including speech-to-text conversion, local storage on the device, and the application's user interface. Team Charlie is making more progress to determine the internal design decisions.
- **Human Interface Design**: This section focuses on how the user interacts and uses Mnemosyne mobile application. The section focuses on basic functionalities such as setting, home, new notes, view notes, and undo. Mainly, it is the last stage where the app is functional and ready to deploy for the user to use.

Reference Material

Table 1 References

Title	Reference Location
Course Material	UMGC SWEN 670 Course Material
10 Usability Heuristics for User Interface Design	https://www.nngroup.com/articles/ten-usability-heuristics/
Cloud Speech-to-Text / Error messages	https://cloud.google.com/speech-to-text/docs/error-messages
Cloud Speech-to-Text / adaptation model	https://cloud.google.com/speech-to-text/docs/adaptation-model
Cloud Speech-to-Text / Section-6	https://cloud.google.com/speech-to-text#section-6
Flutter	https://flutter.dev/

Definitions and Acronyms

Table 2 Definitions and Acronyms

Acronym/Abbreviation	Definition/Reference
TDD	Technical Design Document
PII	Personally Identifiable Information
UI/UX	User Interface Design/User Experience Design
AES	Advanced Encryption Standard
PKCS7	Cryptographic Message Syntax
JSON	JavaScript Object Notation
HTTPS	Hypertext Transfer Protocol Secure
API	Application Programming Interface
AI	Artificial Intelligence

2. System Overview

Mnemosyne is a mobile application that records people's voices, especially those with short-term memory disabilities (Alzheimer's). The services of this application focusing on daily speech conversation, whether on a professional level to disabilities level (users). Its whole purpose is to keep track of the last discussion that helps people retrospectively. The use of device (mobile, desktop) user interface record, google cloud, Mnemosyne management creation base on Artificial Intelligent (AI) that interprets user speech recognitions shall be a central core of the final product.

At a high level of breaking down to the service, Team Charlie explored the path of taking advantage of Google Cloud services. Our primary techniques are to ensure the speech is functionally based on the requirements that outline the software requirement specification. Also, the DevSecOps team is involved in providing a service for our team to process data, record, and establish tools that can test our code and audit the entire Mnemosyne system applications.

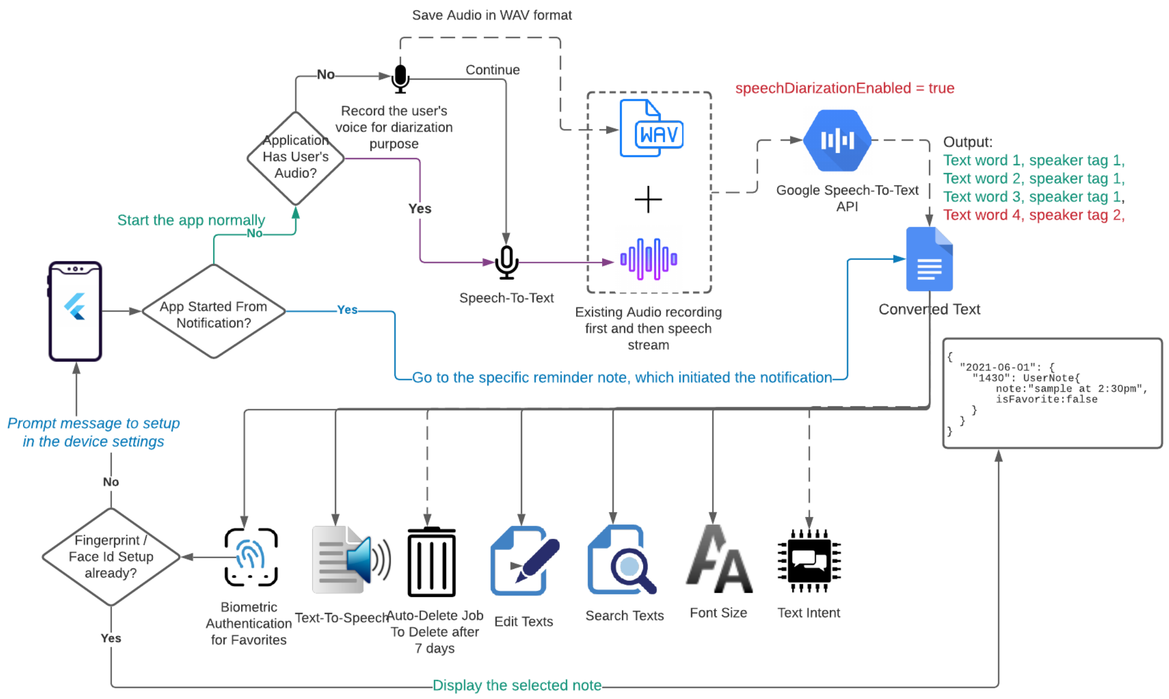
The Mnemosyne architecture consists of 6 main core components: Speech to Text API, Data Persistence, BatchDelete Job, Text to speech Converter, Data Security, and Speaker Diarization. Refer to the system architecture section for a detailed explanation.

3. System Architecture

The following section will elaborate on the high-level system architecture of the application under development.

3.1 Architectural Design

The Mnemosyne application is intended to be hosted on the local device, with some of the components taking advantage of Google cloud services. The major components that make the application work from end to end can be defined as following: audio streaming from the device, speech-to-text conversion, data persistence in the local device, batch delete of old data, text-to-speech, PII security, user interface, and voice training to distinguish the user from other speakers. Each of these components will be elaborated on in detail in the following sections.



Save Audio in WAV format
No
Continue
Record the user's voice for diarization purpose
Application Has User's Audio?
Yes
speechDiarizationEnabled = true
WAV
Google Speech-To-Text API
Output:
Text word 1, speaker tag 1,
Text word 2, speaker tag 1,
Text word 3, speaker tag 1,
Text word 4, speaker tag 2,
Start the app normally
No
App Started From Notification?
Yes
Speech-To-Text
Existing Audio recording first and then speech stream
Converted Text
Go to the specific reminder note, which initiated the notification
Prompt message to setup in the device settings
No
Fingerprint / Face Id Setup already?
Yes
Biometric Authentication for Favorites
Text-To-Speech
Auto-Delete Job To Delete after 7 days
Edit Texts
Search Texts
Font Size
Text Intent
Display the selected note

```
{
  "2021-06-01": {
    "1430": UserNote{
        note:"sample at 2:30pm",
        isFavorite:false
    }
  }
}
```

As shown in the above architecture design, the components are like below.

Step	Process	Error Handling
1	Speech-to-text API · User activates the application and starts speaking. · Google clouds Speech-to-text service	If error, then display an error message
2	Data persistence · Local file in JSON format · Map of < date, map of <Time, Text>>	If error, then let the end-user know
3	BatchDelete of the data older than a set number of days	
4	Text-to-Speech converter	If error, then let the end-user know
5	Security · PII Encryption · Biometric key	
6	Speaker diarization	

	· Voice training to enable the system to distinguish the user from other speakers	
7	Text intent · Parse the text . Save for notification	
8	Notification · flutter_local_notification for notification	

3.2 Decomposition Description

As mentioned in the high-level component description above, each component makes the application work from end to end. In the following sections, each of the components is described with more technical details.

3.2.1 Speech-To-Text Service

Mnemosyne app deals mainly with speech-to-text and text-to-speech. The core component of the application, the Speech-To-Text service, is architected to have the best experience with high performance. The application will be designed to constantly stream the audio data from Mic (Microphone) as the application is activated. The data is then posted to Google's speech-to-text API through a Service Account. Service Accounts are nothing but temporary access management roles to the cloud from external services (Mnemosyne). Speech-to-text API is a managed service of Google, which supports both audio files and audio streams. The API converts the data to text and streams it back constantly over HTTPS protocol. The flutter application collects each data byte and saves it in the local storage.

This service addresses requirements.

- 1.) The application shall not save any voice recording.
- 3.) The application shall provide the option to convert the user's speech to text.
- 16.) The application shall provide the following means to activate recording: Tap on the app and immediate voice recognition.
- 20.) The application shall provide a trigger to end the voice recording.

In addition, using Google's Speech-To-Text API gives the real-time data conversion experience. The only drawback of this design is the application requires the internet to be working.

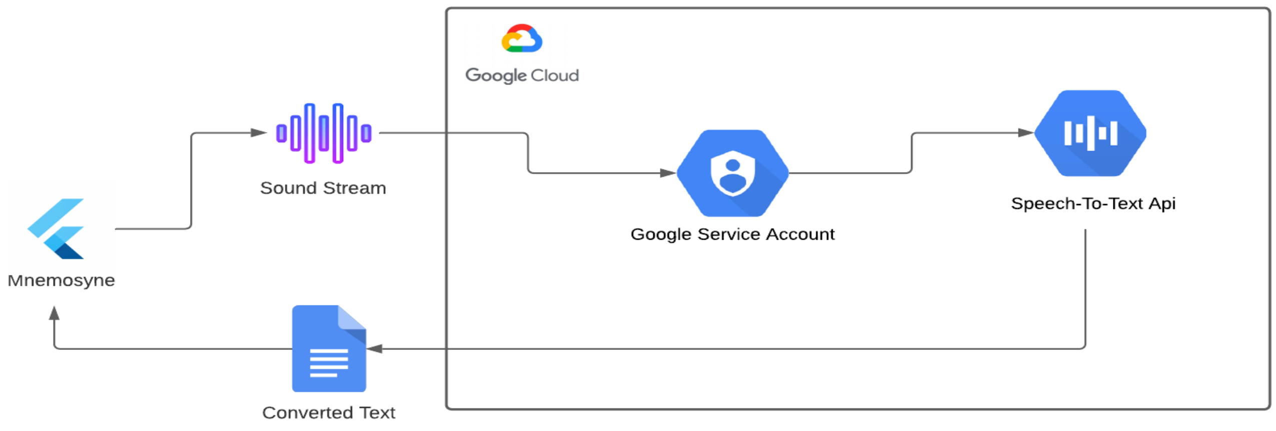


Figure 3.2.1.1 Speech to Text Service

3.2.2 Data Persistence

As Speech-To-Text API converts the speech data to text, it is sent back to the requester, the Mnemosyne application itself. The consumer of the above service is the data persistence module in the application. This service saves the data in JSON format in a file in the local storage. The JSON design in a map format with the key as the date and value is another map. The secondary map will have the key as the time and value as the converted texts. This design helps keep the data segregated by date/time and helps in displaying the same in the user interface. It allows the users to go back to the text for a specific date or a particular time. There are many advantages of saving this data in the local storage, such as data security, high performance, and no internet dependency.

This design addresses the below requirements.

- 14.) The application shall provide the ability to save the speech from converting texts to local device storage.

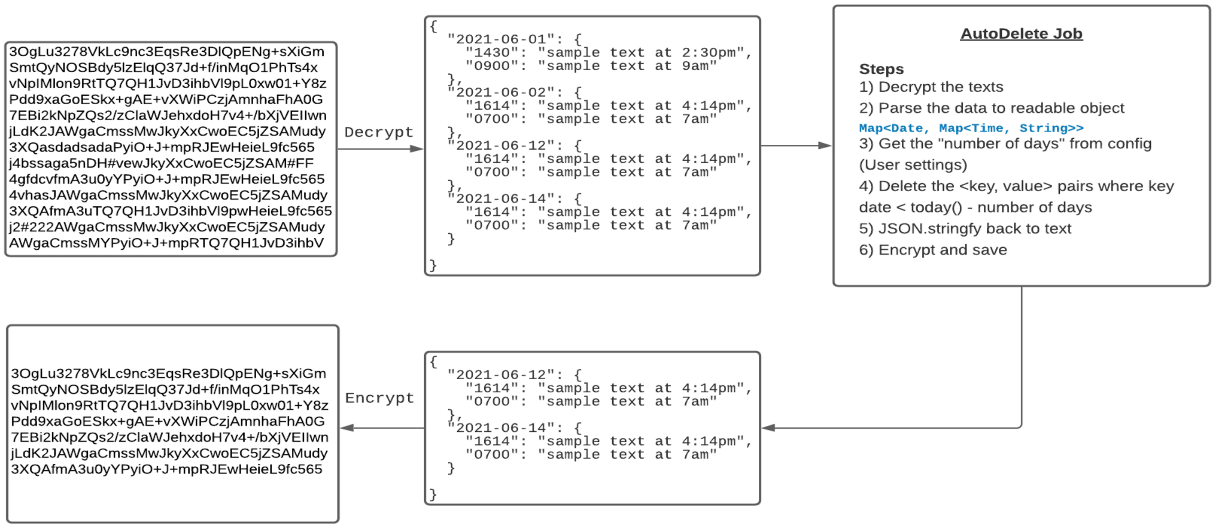
3.2.3 BatchDelete Job

The application will take advantage of background task jobs that will delete or purge the older than specified days. The default number of days will be set to 7. This job goes through the data decryption and encryption process of the text to purge the data. As the data is saved in an encrypted format, the data must be decrypted before parsing it into readable texts. As the data is passed back to the JSON format, the job will remove the data older than the specified number of days. As the data is purged, it will be saved back to the local storage after going through the encryption process. The below figure describes the steps taken in the job.

This job addresses the below requirements.

- 19.) The application shall retain speech to text recognition notes for one week in duration.

The only drawback of this job is, it will do a hard delete of the eligible data, which may not be recoverable anymore. In case of a user error, the data will be removed from the local storage.



```
3OgLu3278VkLc9nc3EqsRe3DlQpENg+sXiGm
SmtQyNOSBdy5lzElqQ37Jd+f/inMqO1PhTs4x
vNpIMlon9RtTQ7QH1JvD3ihbVl9pL0xw01+Y8z
Pdd9xaGoESkx+gAE+vXWiPCzjAmnhaFhA0G
7EBi2kNpZQs2/zClaWJehxdoH7v4+/bXjVEIlwn
jLdK2JAWgaCmssMwJkyXxCwoEC5jZSAMudy
3XQasdadsadaPyiO+J+mpRJEwHeieL9fc565
j4bssaga5nDH#vewJkyXxCwoEC5jZSAM#FF
4gfdcvfmA3u0yYPyiO+J+mpRJEwHeieL9fc565
4vhasJAWgaCmssMwJkyXxCwoEC5jZSAMudy
3XQAfmA3uTQ7QH1JvD3ihbVl9pwHeieL9fc565
j2#222AWgaCmssMwJkyXxCwoEC5jZSAMudy
AWgaCmssMYPyiO+J+mpRTQ7QH1JvD3ihbV
```

```
{
  "2021-06-01": {
    "1430": "sample text at 2:30pm",
    "0900": "sample text at 9am"
  },
  "2021-06-02": {
    "1614": "sample text at 4:14pm",
    "0700": "sample text at 7am"
  },
  "2021-06-12": {
    "1614": "sample text at 4:14pm",
    "0700": "sample text at 7am"
  },
  "2021-06-14": {
    "1614": "sample text at 4:14pm",
    "0700": "sample text at 7am"
  }

}
```

Steps
1) Decrypt the texts
2) Parse the data to readable object
`Map<Date, Map<Time, String>>`
3) Get the "number of days" from config (User settings)
4) Delete the <key, value> pairs where key date < today() - number of days
5) JSON.stringfy back to text
6) Encrypt and save

```
{
  "2021-06-12": {
    "1614": "sample text at 4:14pm",
    "0700": "sample text at 7am"
  },
  "2021-06-14": {
    "1614": "sample text at 4:14pm",
    "0700": "sample text at 7am"
  }

}
```

```
3OgLu3278VkLc9nc3EqsRe3DlQpENg+sXiGm
SmtQyNOSBdy5lzElqQ37Jd+f/inMqO1PhTs4x
vNpIMlon9RtTQ7QH1JvD3ihbVl9pL0xw01+Y8z
Pdd9xaGoESkx+gAE+vXWiPCzjAmnhaFhA0G
7EBi2kNpZQs2/zClaWJehxdoH7v4+/bXjVEIlwn
jLdK2JAWgaCmssMwJkyXxCwoEC5jZSAMudy
3XQAfmA3u0yYPyiO+J+mpRJEwHeieL9fc565
```

The subsequent releases will have more features to archive the data for additional days as a buffer that can only assess demand.

Figure 3.2.3.1 – Batch Delete

3.2.4 Text-to-Speech

The accessibility features include notes that will have an option to read the texts. The speaker icon will trigger an API call to convert the text back, as displayed in the above figure. For this, the application will take advantage of the Text-to-Speech library instead of any cloud services.

This text-to-speech service covers the below part of the requirement.

- 5.) The application should read the text and play the synthesized speech into the user's earbuds.

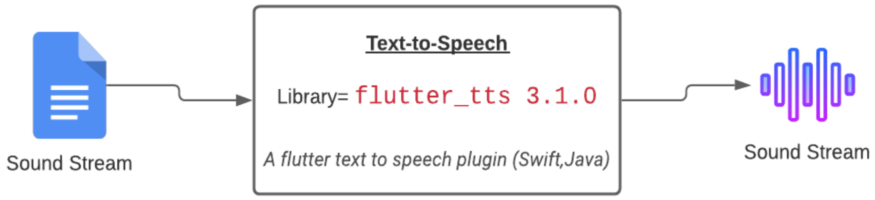

Sound Stream
Text-to-Speech
Library= flutter_tts 3.1.0
A flutter text to speech plugin (Swift,Java)
Sound Stream

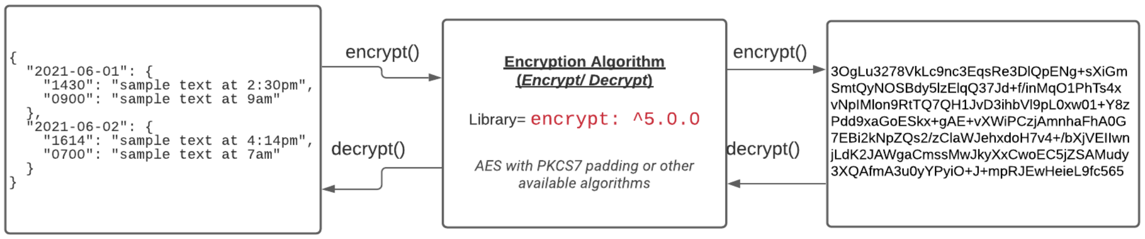
Figure 2.2.5.1 – Text-to-Speech

3.2.5 Data Security

Since the application revolves around user activity and records personal information, data security must be considered part of the application development. Since the data lives in the device storage itself, it can be encrypted while data at rest. The encryption is going to be done using the flutter library called "encrypt." The library supports many algorithms. AES with PKCS7 padding will be used for this purpose. The key for encryption is going to be the user agent id or the device id. Additional biometrics authentication adds to make the data more secured and accessed by the user.

This section covers the below requirements.

- 22.) The application must encrypt the data at rest.



```
{
  "2021-06-01": {
    "1430": "sample text at 2:30pm",
    "0900": "sample text at 9am"
  },
  "2021-06-02": {
    "1614": "sample text at 4:14pm",
    "0700": "sample text at 7am"
  }
}
```

```
3OgLu3278VkLc9nc3EqsRe3DlQpENg+sXiGm
SmtQyNOSBdy5lzElqQ37Jd+f/inMqO1PhTs4x
vNpIMlon9RtTQ7QH1JvD3ihbVl9pL0xw01+Y8z
Pdd9xaGoESkx+gAE+vXWiPCzjAmnhaFhA0G
7EBi2kNpZQs2/zClaWJehxdoH7v4+/bXjVEIlwn
jLdK2JAWgaCmssMwJkyXxCwoEC5jZSAMudy
3XQAfmA3u0yYPyiO+J+mpRJEwHeieL9fc565
```

Figure 2.2.5.1 – Encryption

3.2.6 Speaker Diarization

Mnemosyne application will be able to capture the user's voice only for the conversion to text. Since the user may get involved in the conversations and activate the application while multiple people are speaking, the system is expected to stream the user's voice. For this, the application will take advantage of Google's Speech-to-Text configurations and options. As part of the API request, the flag "enableSpeakerDiarization" will be set to true. As part of the response, the API will send back the texts with the "speaker tag" number. The text strings will be segregated based on the speaker tag. This way user will not ask for permission to record anyone else's voice.

This part of the design will cover the below requirements.

- 11.) The application shall ignore everything except what the user speaks.
- 13.) The application shall bypass asking everyone permission to record.
- 12.) The application shall provide the means for the user to train the app on its voice.

3.2.7 Text Intent

This application will use a feature to parse the saved text and identify the texts that has a scheduled text. This is done by using the text parsing feature available in the dart language. The keywords like ‘a.m.’, ‘p.m.’, ‘o clock’, ‘tonight’, ‘tomorrow’ are saved as standard keywords. However, user will be given an option to save additional keywords, that they want to identify and get notified on. Apart from that, the texts with confidential information like SSN, medicare id, etc will also be identified as the keywords. These words will be saved as favorite automatically.

3.2.8 App Notifications

Mnemosyne app will take the most advantage of the flutter libraries to notify the user on a schedule or immediately. During the development of the application, the platform compatibility will be under consideration, and ensured the app notifies as expected on all operating systems.

3.3 Exception Handling & Design Rationale

a. Exception Handling

- Exception handling within the API manages streamed audio, allowing the system to process speech transcription into text safely. Speech adaptation, model adaptation, and enabling word-level confidence are features that help avoid critical failures, speech accuracy, and other failures. API's manage exception handling in the cloud and return applicable error messages to the application. Speech-to-Text errors will be processed by API exception handling.

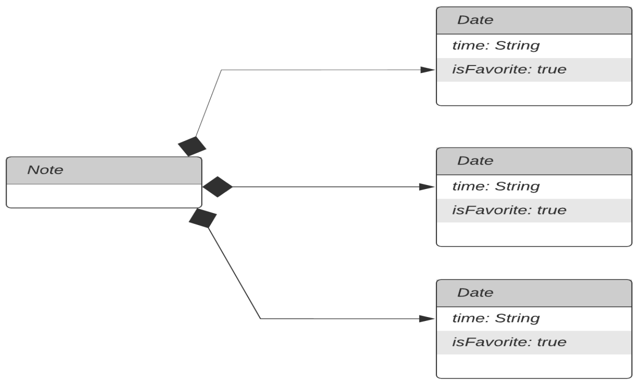
b. Design Rationale

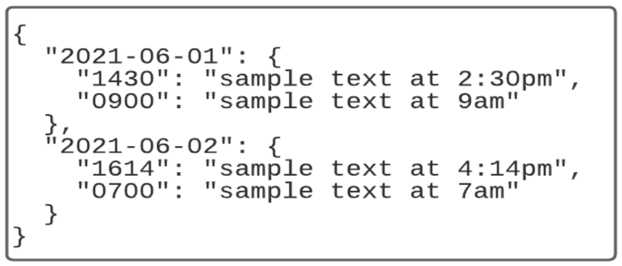
- The design rationale is forming through utilizing solutions configured within the boundaries of the project, technical, and functional requirements.
- Google Speech-to-Text API: Focus being a mobile application, the Speech-to-Text had multiple design features that met the requirements. Standard models are free for personal systems like phones and tablets. The API also seemed more accessible for the development team. Additionally, Google provides options to recognize distinct speakers in a conversation, which can utilize in the application under development.
- Sound Stream: To continuously stream the audio from the device to the external API, a flutter library called 'sound stream' was selected to use.

- Application Layer: The application layer utilizes a dart-based Flutter framework. Flutter is a Google SDK for building natively compiled applications for multiple devices from a single codebase. Flutter is free and open source. It was determined by the primary stakeholder to develop the application with the flutter framework.
- Back-End: It is JSON format, which is speech to text in notes by date and time. The file is encrypted with a retention policy of 7 days, managed by a work manager as stated in the requirements.

4. Data Design

These are the currently proposed data entities that will get stored locally or remotely by the application. From these relationships, critical information can be identified and held by the user.





```
{
   "2021-06-01": {
      "1430": "sample text at 2:30pm",
      "0900": "sample text at 9am"
   },
   "2021-06-02": {
      "1614": "sample text at 4:14pm",
      "0700": "sample text at 7am"
   }
}
```

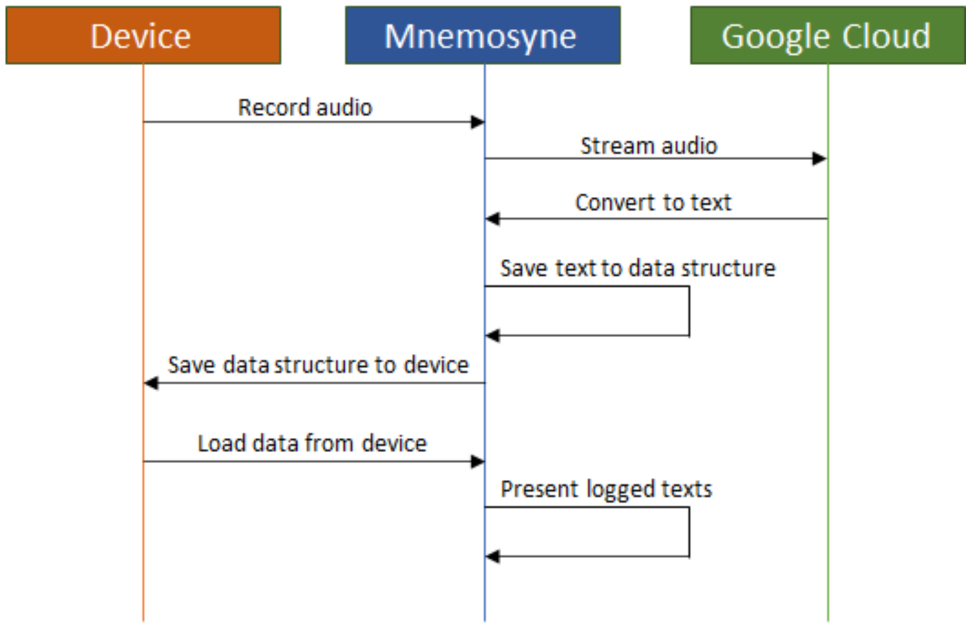
Figure 2.2.2.1 – Data structure

Note	Field	Type	NULL	Default
	Date	date		
	time	time		
	text	string		
	isFavorite	bool		false

5. Component Design

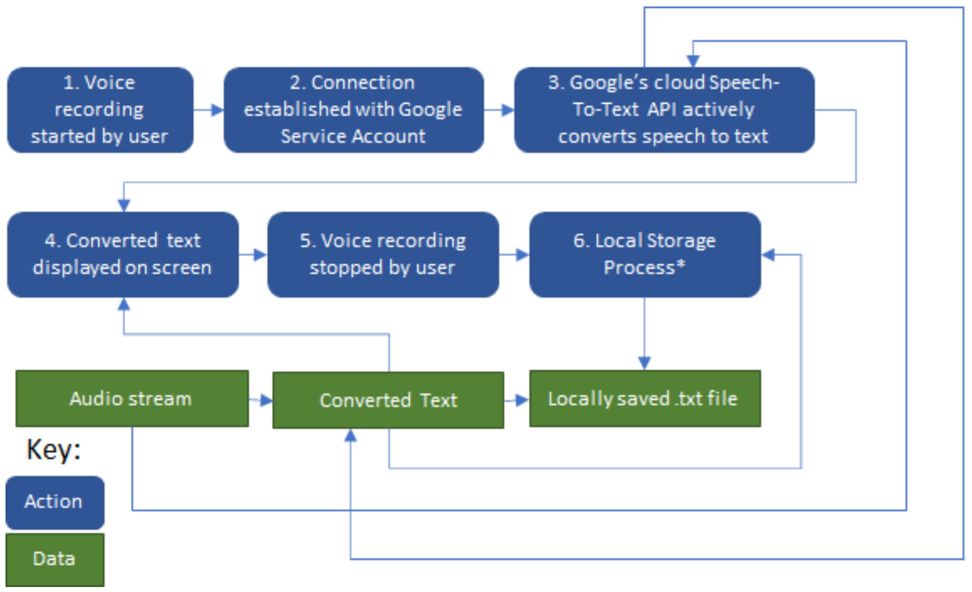
5.1 Overall Design

As described in the System Architecture section, the application utilizes a few significant components that communicate with one another to make the application functional. These components broadly include speech-to-text conversion, local storage on the device, and the application's user interface.



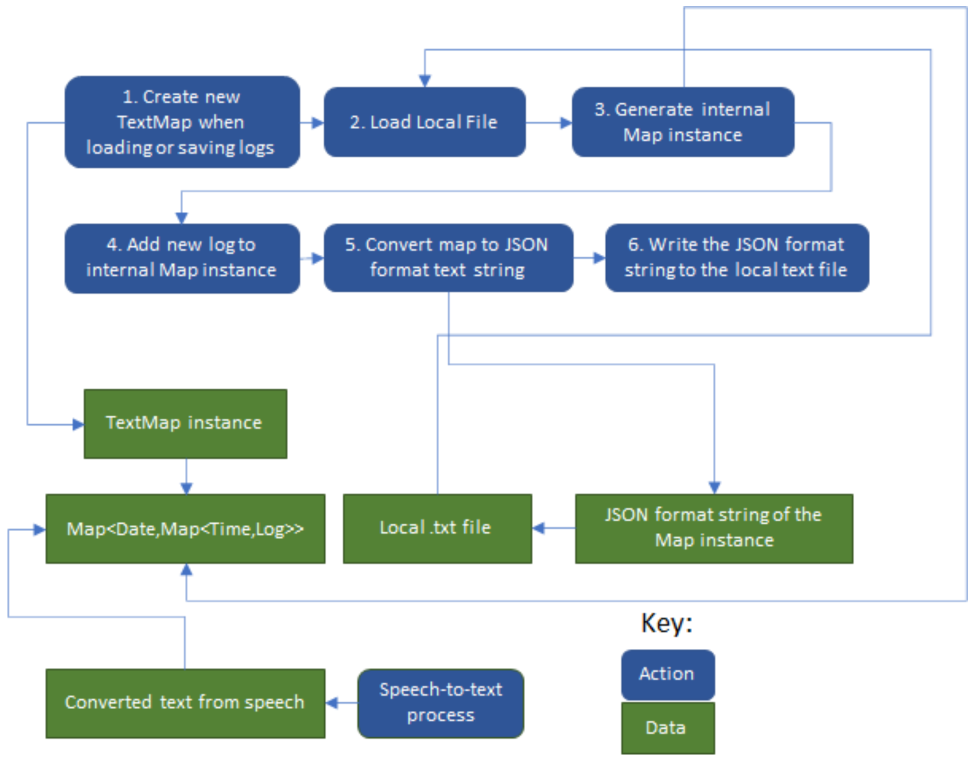
5.2 Speech-to-Text

The conversion of speech to text is accomplished by taking advantage of Google cloud services, which significantly support Flutter/Dart development.


1. Voice recording started by user
2. Connection established with Google Service Account
3. Google's cloud Speech-To-Text API actively converts speech to text
4. Converted text displayed on screen
5. Voice recording stopped by user
6. Local Storage Process*
Audio stream
Converted Text
Locally saved .txt file
Key:
Action
Data

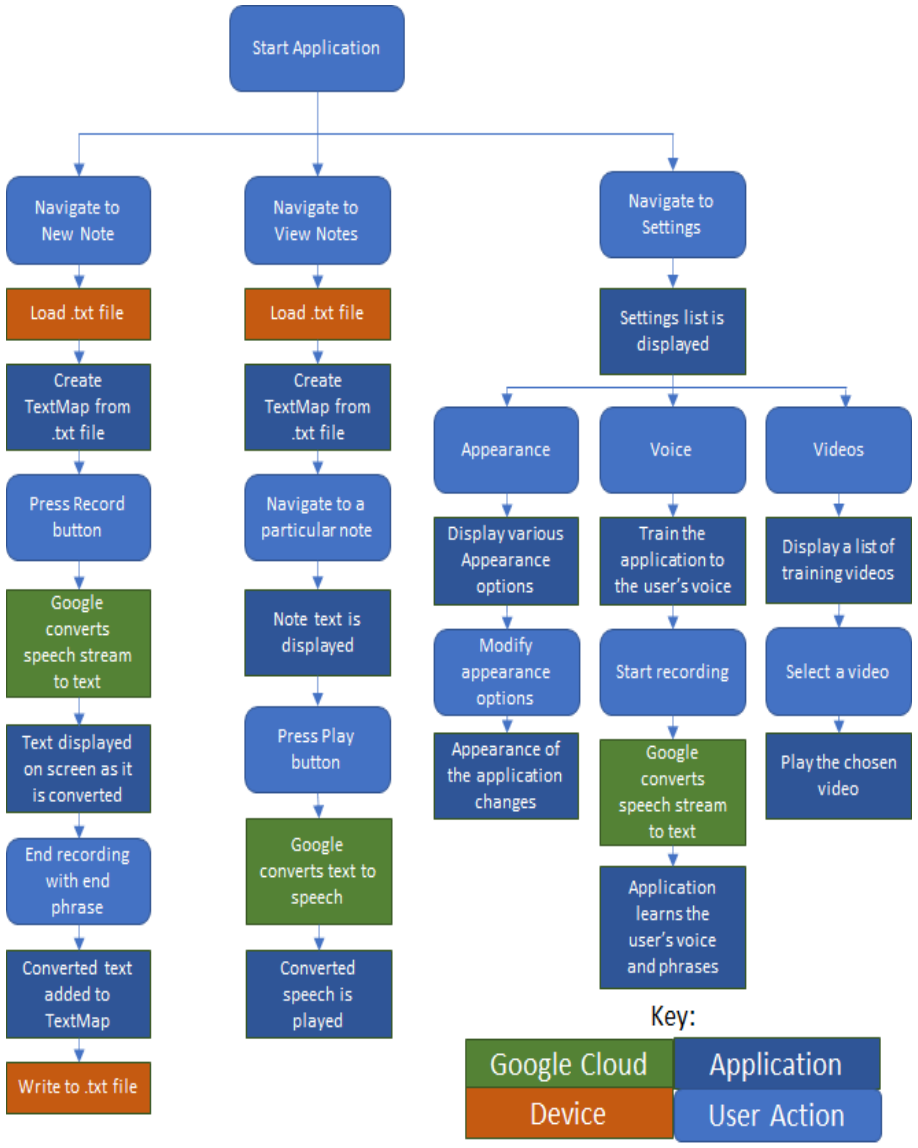
5.3 Local Storage

Once the speech converts to text that the application can understand, this text is saved locally on the device. Details on these processes and data structures can be found in other sections of this document.


1. Create new TextMap when loading or saving logs
2. Load Local File
3. Generate internal Map instance
4. Add new log to internal Map instance
5. Convert map to JSON format text string
6. Write the JSON format string to the local text file
TextMap instance
Map<Date,Map<Time,Log>>
Local .txt file
JSON format string of the Map instance
Converted text from speech
Speech-to-text process
Key:
Action
Data

When just reading the saved notes instead of writing a new one, only the first three steps are necessary.

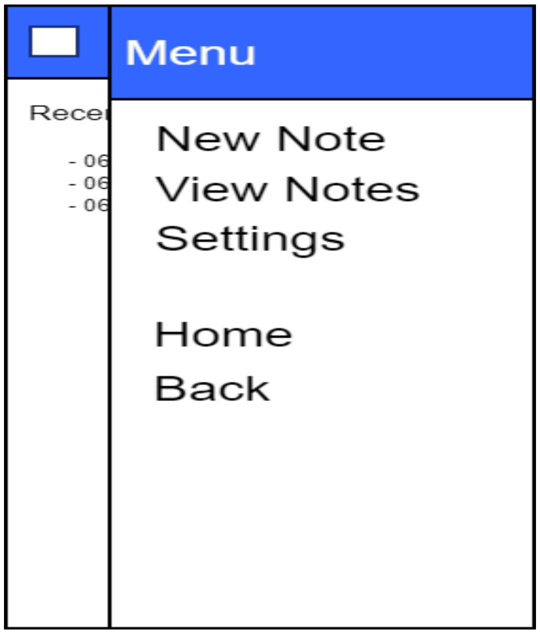
5.4 Flow Diagram


Start Application
Navigate to New Note
Load .txt file
Create TextMap from .txt file
Press Record button
Google converts speech stream to text
Text displayed on screen as it is converted
End recording with end phrase
Converted text added to TextMap
Write to .txt file
Navigate to View Notes
Load .txt file
Create TextMap from .txt file
Navigate to a particular note
Note text is displayed
Press Play button
Google converts text to speech
Converted speech is played
Navigate to Settings
Settings list is displayed
Appearance
Display various Appearance options
Modify appearance options
Appearance of the application changes
Voice
Train the application to the user's voice
Start recording
Google converts speech stream to text
Application learns the user's voice and phrases
Videos
Display a list of training videos
Select a video
Play the chosen video
Key:
Google Cloud
Application
Device
User Action

6. Human Interface Design

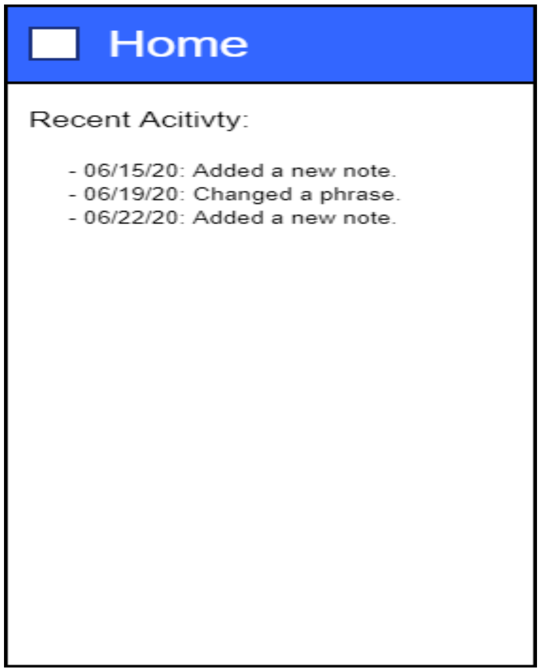
6.1 Overview of User Interface

The user interface aims to be as intuitive and natural as possible while still providing the user effective control over the application. This objective is primarily accomplished with a persistent menu screen that can be opened by pressing the button on the top left of the application.



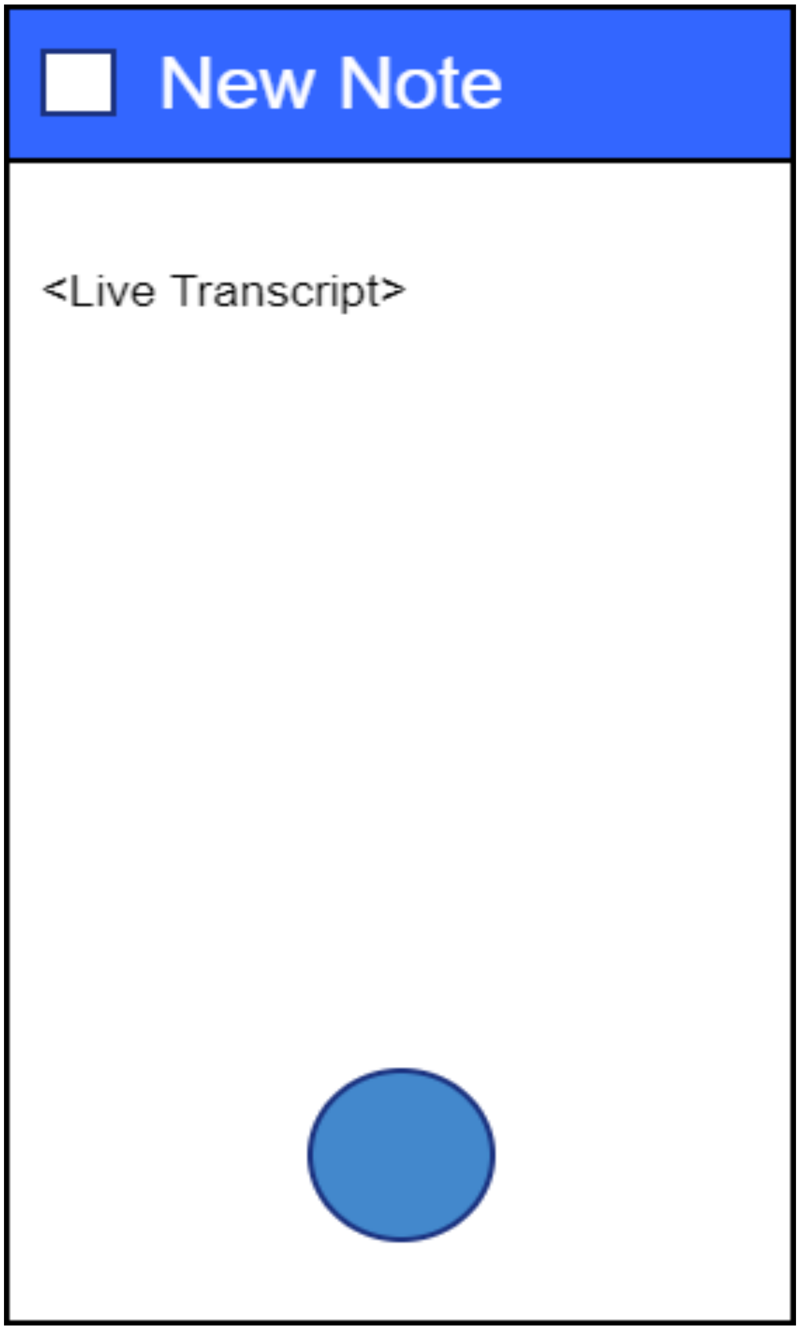
In this spot, the user can navigate to one of the three main screens, which can be further dividing into separate screens depending on the function.

6.2 Home Screen


Home
Recent Acitivty:
- 06/15/20: Added a new note.
- 06/19/20: Changed a phrase.
- 06/22/20: Added a new note.

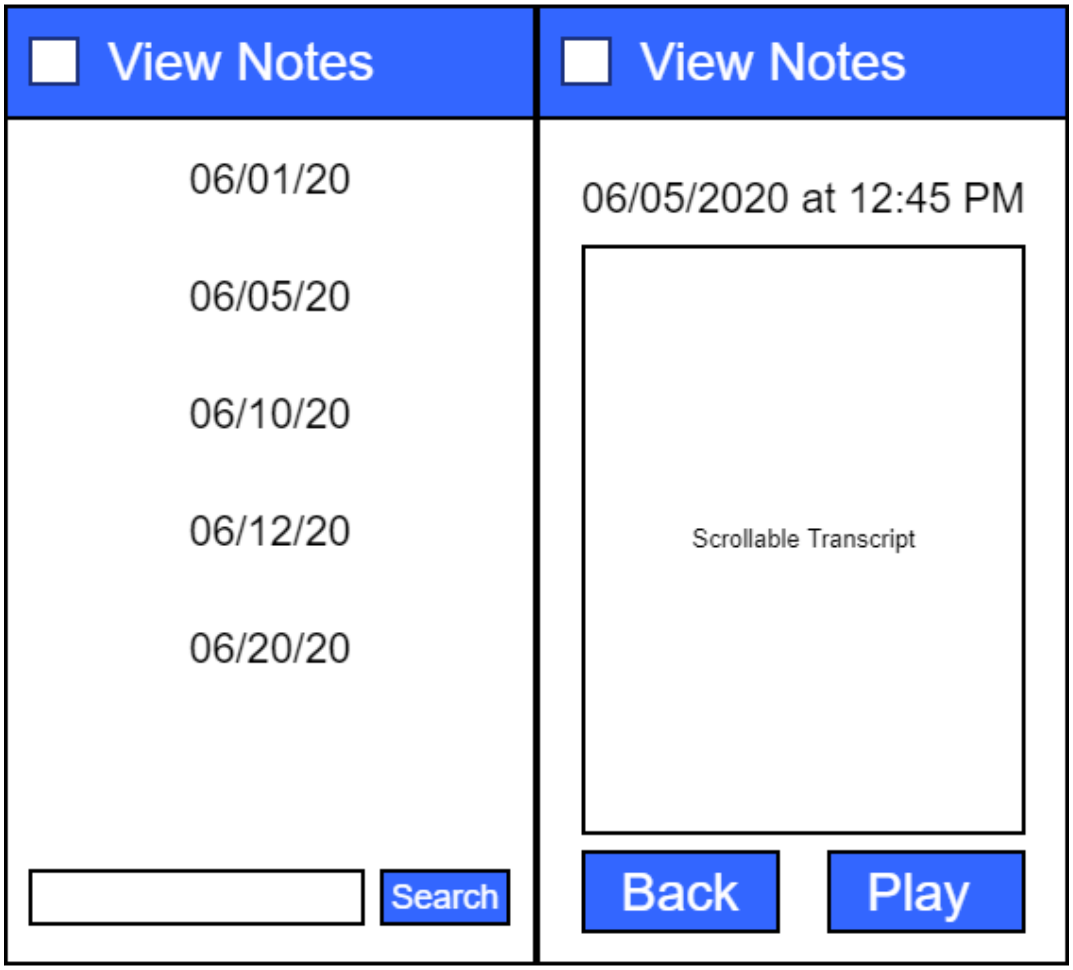
The Home screen serves as a landing pad when the user first opens the application. Recent activities are presented to the user to remind them what they have done in the application on this screen.

6.3 New Note Screen



The New Note screen is a simple screen that allows the user to begin recording their voice by clicking the round button at the bottom. The application will then start listening to recorded audio until the user says one of the configured phrases. When a recording begins, the text of that recording will be displayed in the center of this screen.

6.4 View Notes Screen



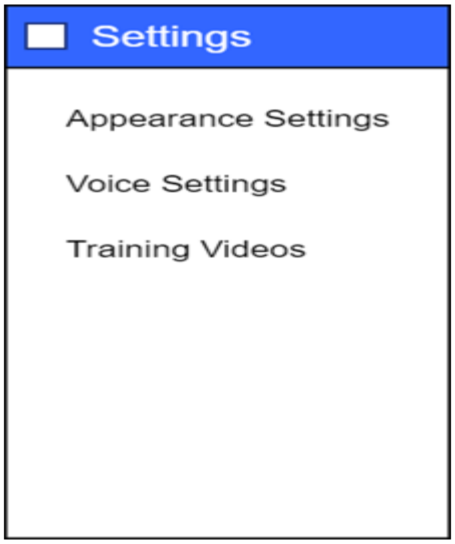
On the View Notes screen, the user can scroll through a list of dates, select one, then a list of times, select one, then view the recorded note for that date and time. Dates and times will only be shown if there is a relevant recording. At the bottom of the screen, the user can type a search term and search through all notes.

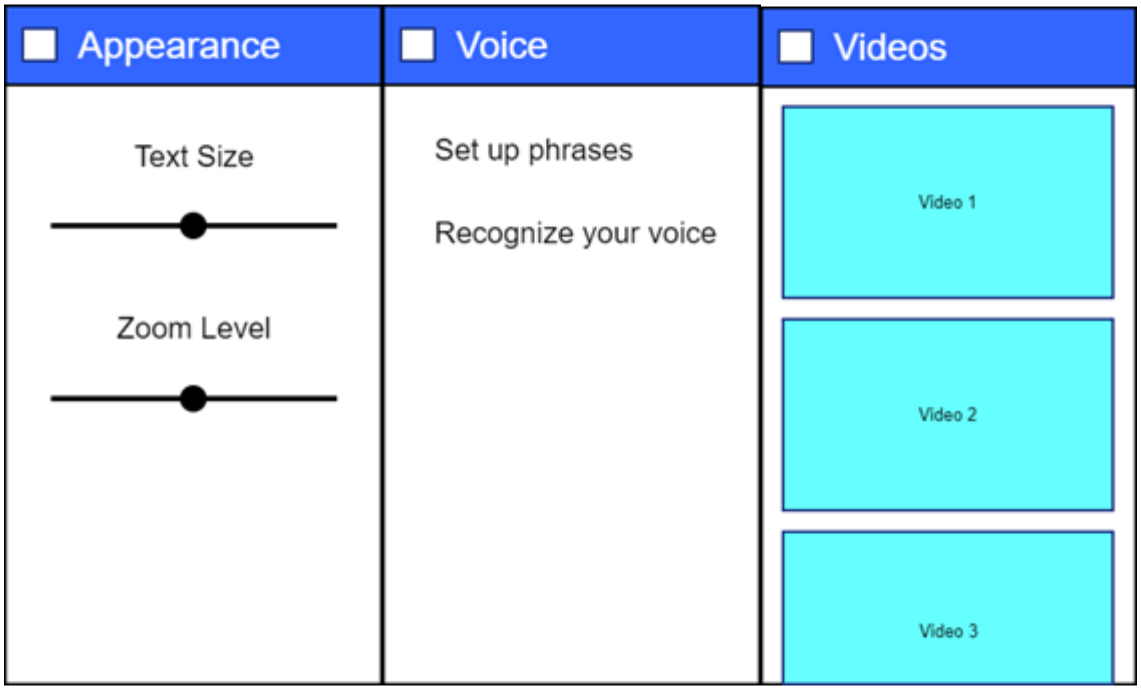
A scrollable text transcript and a Play button will playback the transcript as audio on the actual Note screen. While the user can always go back with the top menu button, there is also a Back button at the bottom of the Note screen for ease of use.

The notes are allowed to be edited by the users addressed the below requirements.

- 18.) The application shall retrieve all results related to the search command.

6.5 Settings Screen





The various Settings screens are used to customize the application and provide video training to the user. The customizations include text size, zoom level, and training the application to recognize a particular user's voice and custom phrases to begin transcribing.

The following requirements are covered by this UI design.

- 4.) The application shall provide a user interface that incorporates the following device features: Bold Text, Display Zoom, Increase Text Size, and High Contrast Colors. (Optional)
- 7.) The application shall provide training videos within the app to guide the user regarding its various features and functionalities.

7. Requirement Matrix

Requirement	Covered in TDD
1.) The application shall not save any voice recording.	3.2.1 Speech-to-Text
2.) The application shall allow the user to edit any speech converted to text.	3.2.4.1 Notes View
3.) The application shall provide the option to convert the user's speech to text.	3.2.1 Speech-to-Text
4.) The application shall provide a user interface that incorporates the following device features: Bold Text, Display Zoom, Increase Text Size, and High Contrast Colors. (Optional)	6.5 Settings Screen
5.) The application should read the text and play the synthesized speech into the user's earbuds.	3.2.5 Text-to-Speech
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.	6. Human Interface Design
7.) The application shall provide training videos within the app to guide the user regarding its various features and functionalities.	6.5 Settings Screen
8.) The application shall recognize distinct phrases and sentences that he or she uses while speaking to him or herself or with others.	3.6 Speaker Diarization
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 the phrases that the user wants to use speaking to him or herself trying to retrieve the noted information.	3.7 Text Parsing
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 and retrieve saved information to play as speech.	3.7 Text Parsing
11.) The application shall ignore everything except what the user speaks.	3.2.6 Speaker Diarization
12.) The application shall provide the means for the user to train the app on its voice.	3.2.6 Speaker Diarization
13.) The application shall bypass asking everyone permission to record.	3.2.6 Speaker Diarization
14.) The application shall provide the ability to save the speech from converting texts to local device storage.	3.2.2 Data persistence
15.) (Optional) – The Mobile Operating System's Virtual Assistant shall interface directly with the Application by Application Name	
16.) The application shall provide the following means to activate recording: Tap on the app and immediate voice recognition.	3.2.1 Speech-to-Text
17.) The application shall provide the ability to search through the saved speech to text notes via text field and/or voice command (Optional).	6. Human Interface Design
18.) The application shall retrieve all results related to the search command.	6.4 View Notes screen

19.) The application shall retain speech to text recognition notes for one week in duration.	3.2.3 BatchDelete Job
20.) The application shall provide a trigger to end the voice recording.	3.2.1 Speech-to-Text
21.) The application must work with IOS and Android	
22.) The application must encrypt the data at rest.	3.2.5 Data Security
23.) The application shall provide a notification banner as a daily event presented to the user (That the application is available and should get used)	3.3.8 Application notification