Technical Design Document – Milestone 2

Team: charlie

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

SWEN 670 – sOFTWARE eNGINEERING pROJECT

June 11, 2021

reVision 1.0

Project name: Mnemosyne, Disability Mobile Application

Date: June 11, 2021

Project Leader: Michael Le

Phase: Design & Engineering and Execution

For approval: Michael Le

Michael le Date: 06/08/2021

For approval: Dr. Mir Mohammed Assadullah

Date: 06/11/2021

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version Number | Date | Description | Approved By |
| 1.0 | 06/11/2021 | Initial Technical Design Document | Michael Le |
|  |  | . |  |
|  |  |  |  |

Table of Contents

[1. Introduction 5](#_Toc66722963)

[1.1 Purpose 9](#_Toc66722965)

[1.2 Scope 12](#_Toc66722966)

[1.3 Overview 13](#_Toc66722967)

[1.4 Reference Material 13](#_Toc66722968)

[1.5 Definitions and Acronyms 14](#_Toc66722969)

[2. System Overview 6](#_Toc66722964)

[2.1 Project Scope 9](#_Toc66722965)

[3. System Architecture 16](#_Toc66722971)

[3.1 Architectural Desgin 16](#_Toc66722972)

[3.2 Decomposition Description 17](#_Toc66722973)

[3.2.1 Dialogflow 19](#_Toc66722974)

[3.2.2 Architectural Desgin 16](#_Toc66722972)

[3.2.3 Decomposition Description 17](#_Toc66722973)

[3.2.4 Dialogflow 19](#_Toc66722974)

[4. Data Design 20](#_Toc66722975)

[4.1 Data Description 20](#_Toc66722976)

[4.2 Data Dictionary 21](#_Toc66722978)

[5 Component Design 28](#_Toc66722987)

[5.1 Service Layer 28](#_Toc66722988)

[5.2 REST Controller 29](#_Toc66722989)

[5.3 AI Component 29](#_Toc66722988)

[6 Human Interface Design 30](#_Toc66722990)

[6.1 Overview of User Interface 30](#_Toc66722991)

[7. Requirement Matrix 50](#_Toc66723017)

[Appendix A – Detail Timeline Word Breakdown Structure 50](#_Toc66723018)

[Appendix B – Project Timeline 53](#_Toc66723020)

# Introduction

**1. Situation sketch and problem definition of the project**

# System Overview

# 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 the advantage of Google cloud services. The major components that make the application work from end to end can be defined as 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 in detail in the following sections.

|  |  |  |
| --- | --- | --- |
| 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 end-user know |
| 3 | BatchDelete of the data older than a set number of days |  |
| 4 | Text-to-Speech converter | If error, then let end-user know |
| 5 | Security  · PII Encryption  · Biometric key |  |
| 6 | Speaker diarization · Voice training to enable to system to distinguish the user from other speakers |  |

## 3.2 Decomposition Description

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

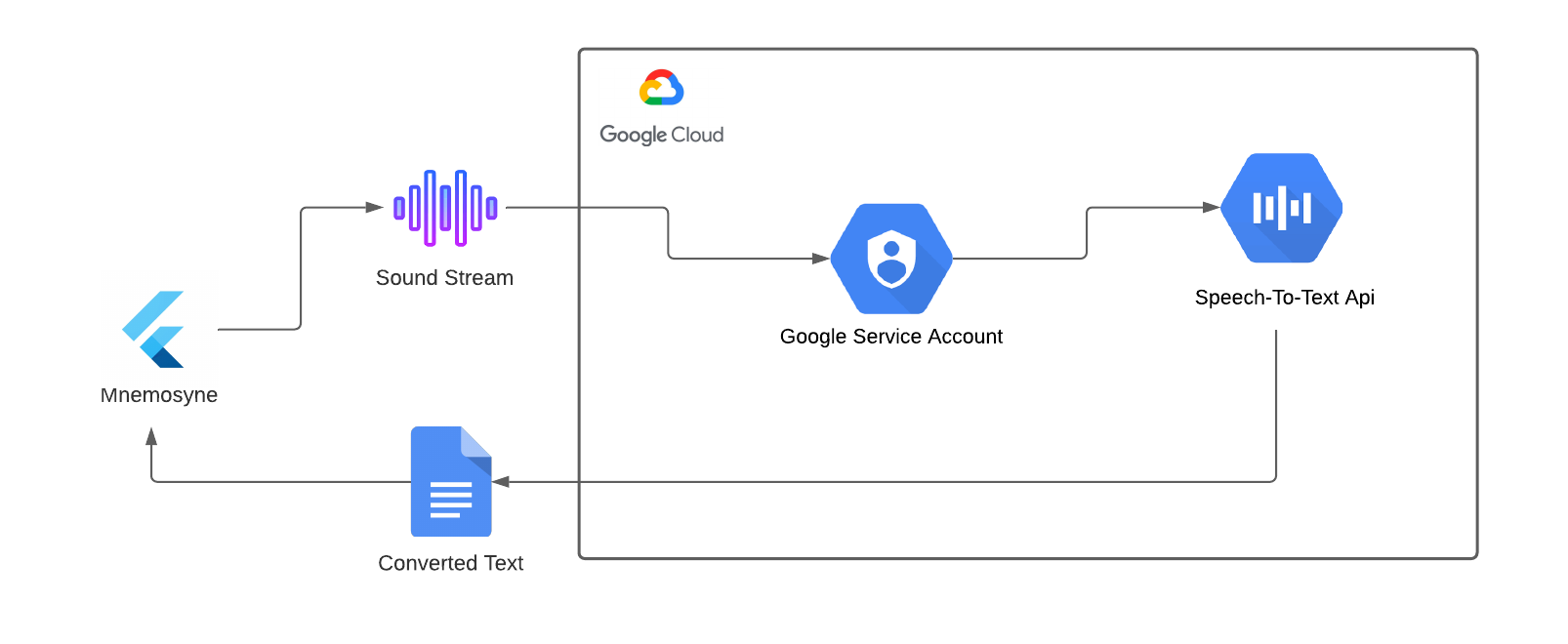
### 3.2.1 Speech-To-Text Service

Mnemosyne app deals mainly around the speech-to-text and vice versa. Being the core component of the application, Speech-To-Text service is architected in such a way that the user will have the best experience with high performance. The application will be designed to constantly stream the audio data from Mic 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 role 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 bytes and saves in the local storage.

This service addresses two major 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. Only drawback of this design is, the application requires 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, which is the Mnemosyne application itself. The consumer of the above service is the data persistence module in the application. This service saves the data in a JSON format in a file in the local storage. The JSON will be designed to be in a map format which will have the key as the date and the value as another map. The secondary map will have the key as the time and value as the converted texts. This design helps to keep the data segregated by date/time and helps in displaying the same in the user interface. This also allows the users to go back to the text for specific date or a specific 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 to conversion texts to local device storage.

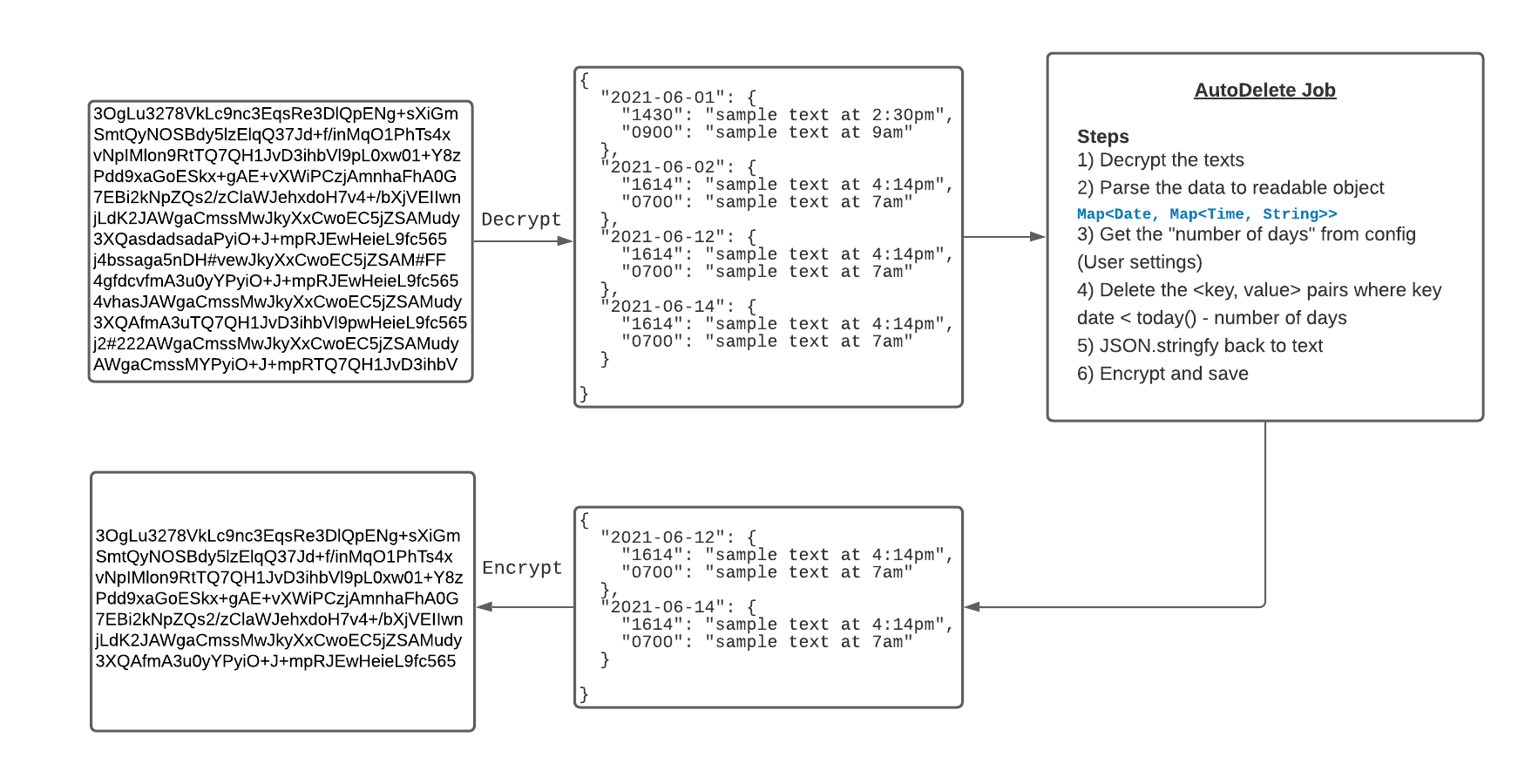
### 3.2.3 BatchDelete Job

The application will take the advantage of background task jobs which will delete or purge the data that are older than the specified number of 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 is required to be decrypted before parsing it to readable texts. As the data is parsed back to the JSON format, the job will remove the data older 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. 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 1 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. The subsequent releases will have more features to archive the data for additional number of days as a buffer which can be accessed only on demand.



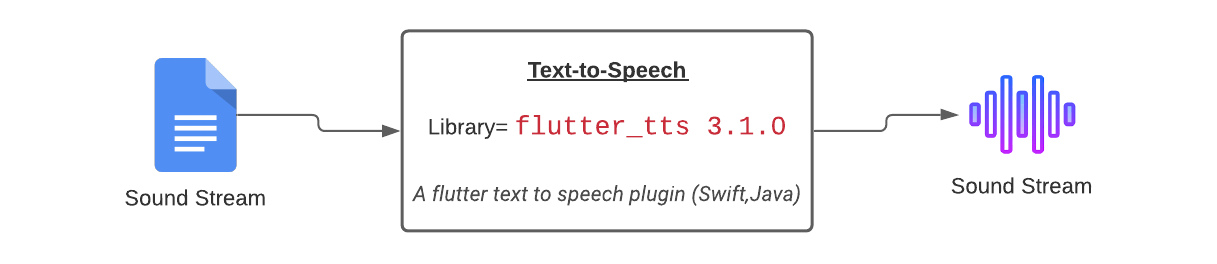
*Figure 3.2.3.1 - Batch Delete*

### 3.2.4 Text-to-Speech)

To have more accessibility features, the notes will have an option to read the texts. As displayed in the above figure, the speaker icon will trigger an API call to convert the text back. For this the application will take the advantage of 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.



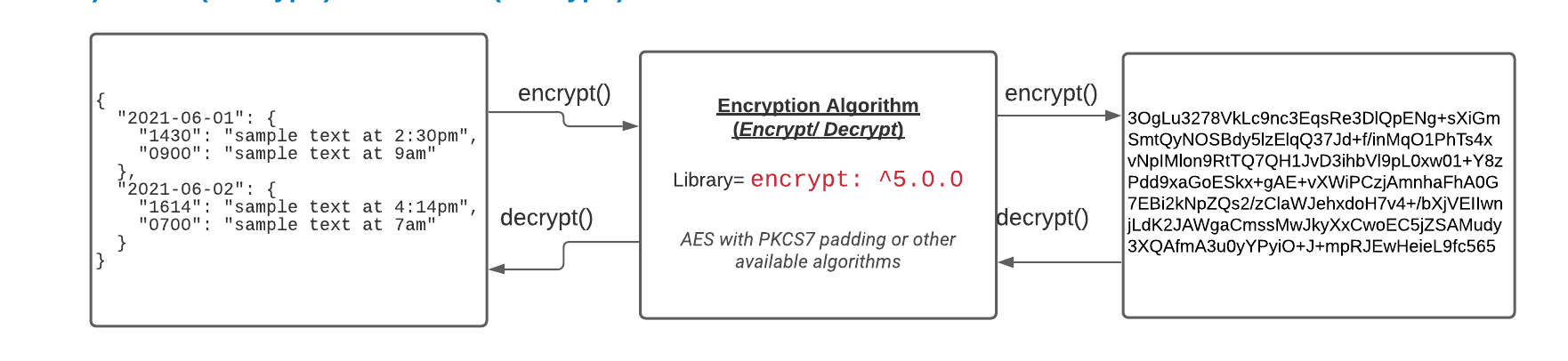
*Figure 2.2.5.1 - Text-to-Speech*

### 3.2.5 Data Security

Since the application revolves a lot around the user activity and records personal information, the data security must be considered as 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 will be added to make the data more secured and can only be accessed by the user.

This section covers the below requirements.

* 22.) The Application must encrypt the data at rest.

*Figure 2.2.5.1 - Encryption*

### 3.2.6 Speaker Diarization

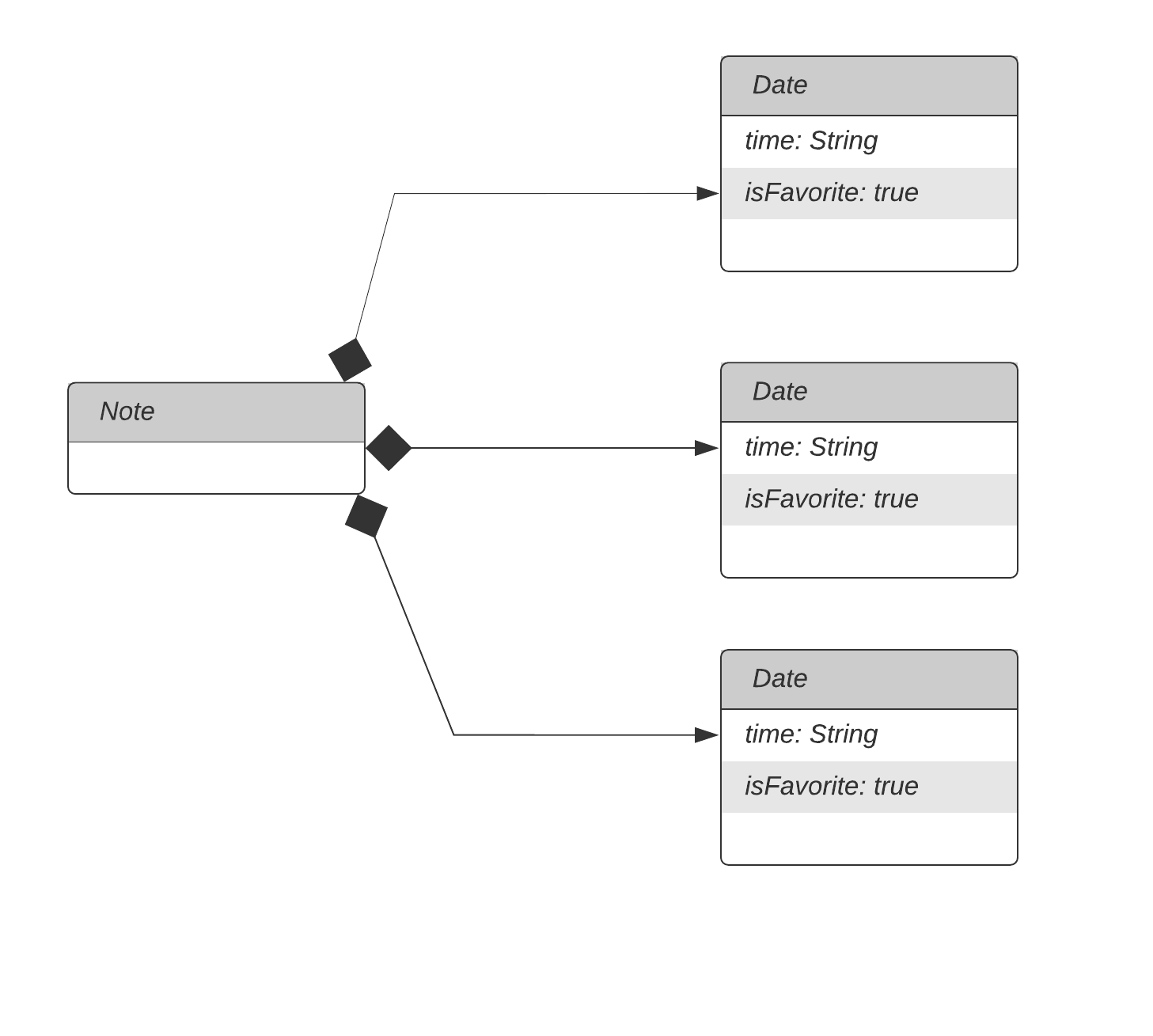
Mnemosyne application will be able to capture the user’s voice only for the conversion to text. Since, the user may get involved in the conversations and activate the application while multiple people are speaking, the system is expected to only stream the user’s voice. For this the application will take the advantage of the 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 “speakerTag” number. The text strings will be segregated based on the speakerTag. This way user will not be asked for the permission to record anyone else’s voice.

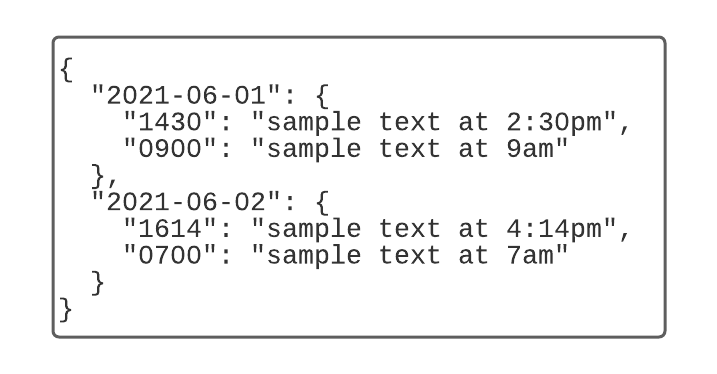
This part of the design will cover the below requirements.

* 11.) The application shall ignore everything except what the user speaks.
* 13.) The application shall bypass asking everyone permission to record.

# Data Design

This is the current proposed data entities that will get stored in locally or remotely by the application. From these relationships key information can be identified and stored by the user.



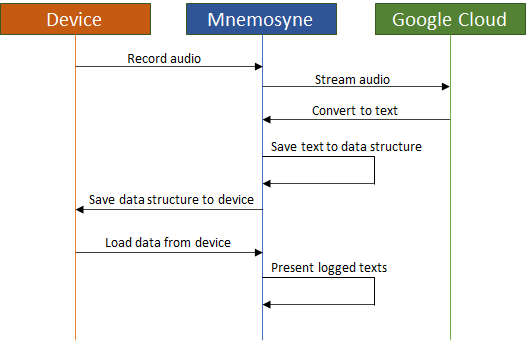


*Figure 2.2.2.1 - Data structure*

# 5. Component Design

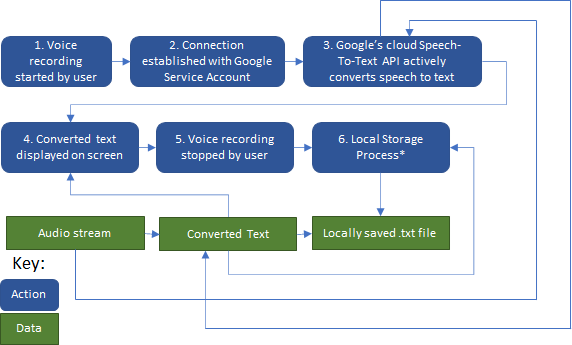
# 5.1 Overall Design

As described in the System Architecture section, the application utilizes a few major 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 user interface of the application.



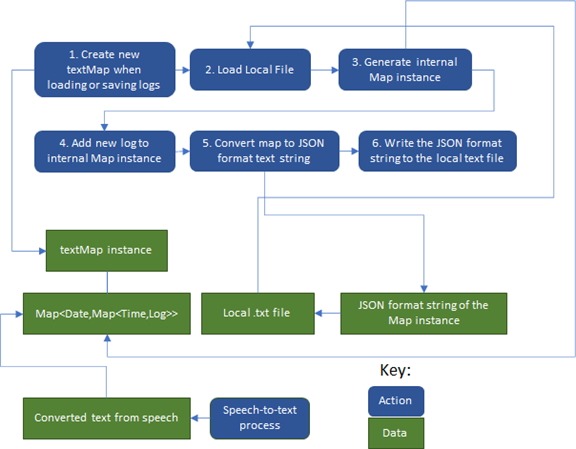
# 5.2 Speech-to-Text

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



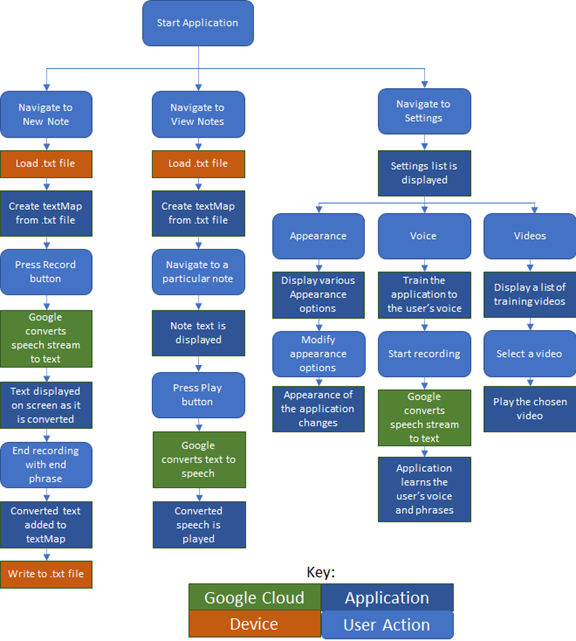
# 5.3 Local Storage

Once the speech has been converted 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.



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

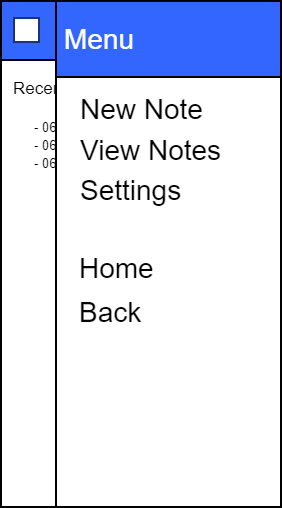
# 5.4 Flow Diagram



# 6. Human Interface Design

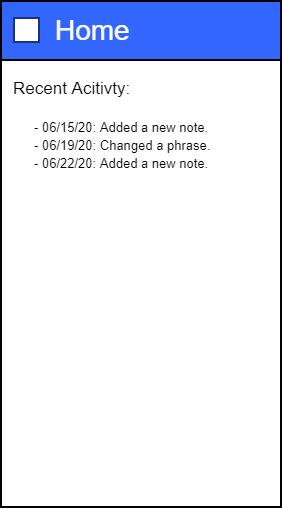
# 6.1 Overview of User Interface

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



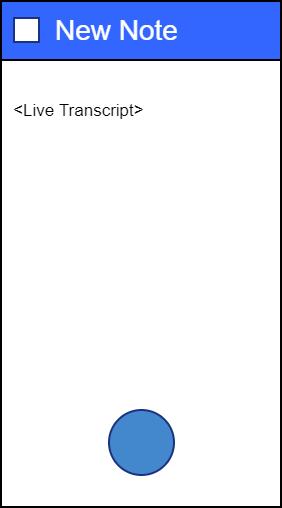
From here the user can navigate to one of the three main screens, which can be further dividing into separate screen depending on the function.

# 6.2 Home Screen



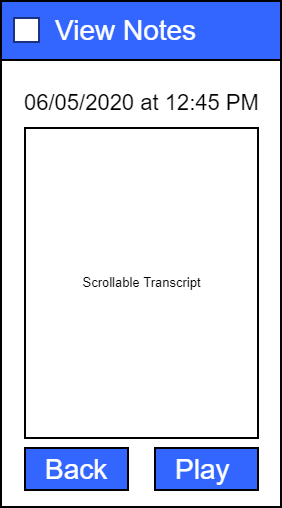
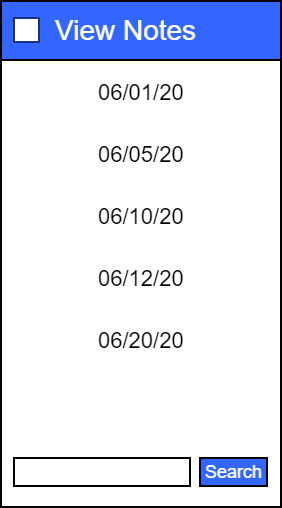
The Home screen serves as a landing pad when the user first opens the application. On this screen, recent activities are presented to the user to remind them what they have done in the application.

# 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 begin 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.

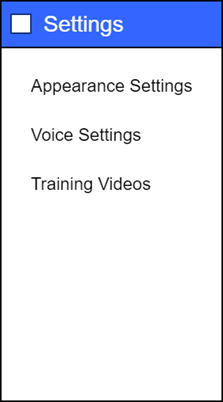
# 6.4 View Notes Screen

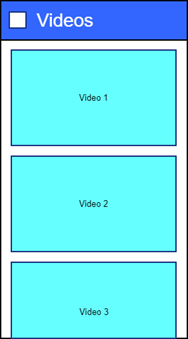
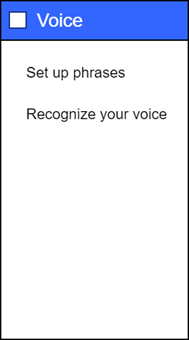
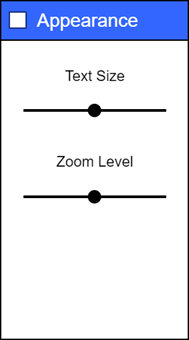


On the View Notes screen, the user can scroll through a list of dates, select one, then a list of times, select one, then view the recorded note for that date and time. Dates and times will only be shown if there is a relevant recording.

On the actual Note screen, there is a scrollable text transcript as well as a Play button that will play back the transcript as audio. 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.

# 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, as well as training the application to recognize a particular user’s voice and custom phrases to begin transcribing.

### 1.1.1 User Interface (Notes View, Search Text, and Settings)

The user interface has many components within it.

#### *1.1.1.1* *Notes View*

This portion of the user interface is the most important part of the application. The hamburger icon on the main screen opens the MenuDrawer which displays the options to view the notes, settings, and other UI options. As the View Notes is selected, the list of notes is displayed with tiles of dates. As the date tiles are selected, the screen displays the notes with the timestamp on it. The notes will be editable by the user.

This view covers the below requirements.

· 2.) The application shall allow the user to edit any speech converted to text.

*Figure 2.2.4.1 - Notes view*

#### *1.1.1.2* *Search Text*

The seatchText service is going to be a lightweight function in

#### *1.1.1.3* *Settings*

The application will have a settings tab, which will contain options to configure the application. It will include options like adding a specific number of days to delete the texts, selecting a voice for the text-to-speech, changing text size, adding catch phrases to highlight in a string of texts.

**Requirement Matrix**