Test Report

Memory Enhancement Application
Milestone 4 –Team Amazing
Software Engineering Project
SWEN 670
August 6, 2021

Version 1.0

Presented by: Shawn Kelly, Christian Ahmed, Mitch Olshansky, Chauntika Broggin, Ayodeji Famudehin, Mo Drammeh, Nicholas Ballo

Milestone 4 – Test Report

Revision History

DATE	VERSION	DESCRIPTION	AUTHOR
08/06/2021	1.0	Initial release	Shawn Kelly

Table of Contents

Table of Figures

Introduction

This report explains the software test process for the Memory Enhancement Application designed and developed by Team Amazing. Memory Enhancement Application is the first version of speech-to-text mobile application that is developed to provide memory reminder support to users through the speech recognition capabilities of mobile phones. To measure and ensure the overall quality of the developed application for its correctness, completeness, usability and performance, the following testing were performed.

Functional testing: It checks the functional requirements of the system.

Non-functional testing: This was used to check all the specified non-functional requirements such as performance, usability, scalability of the developed software.

Regression testing: This was done after codes were fixed for the discovered error. It was responsible for any code upgrades or any other maintenance of the system. It was also utilized to test whether the new code has not affected the existing functionality.

End to End testing: This was used to test the application's workflow from beginning to end. This method basically aimed to replicate real user scenarios to validate the application for integration and data integrity.

1.1 Purpose

The purpose is to evaluate the system for bugs and to check if the developed system satisfies all its stated requirements. This test report represents the organized summary of testing objective, activities, and test result for Memory Enhancement Version 1.0 application. This version of Memory Enhancement application is developed based on both initial and additional Requirements from the User and it is imperative to test the application as per the requirements identified in the application analysis and design phase. To reiterate, the main purpose of this report is to assist all the stakeholders to understand the application quality and decide whether the application, features or defect resolution are on track for the final release of the application to use.

1.2 Application Overview

The design and development of speech-to-text application to cater for the needs of dementia individual had been seen to be highly demanding in recent time. Memory Enhancement application is a smartphone-based speech-to-text system that uses speech recognition technologies to capture user's voice. The audio is transcribed into notes in the middle of conversation using the system's trigger words. This information is saved and encrypted on the system while being accessible to the user by category, verbal recall, and search.

In the design and development of this application, a voice recognition module built into the system performs the speech recognition task while a dedicated android application does the speech transcription. The saved voice is transcribed to English sounds through the android application and matched to the system in-built trigger words to start/stop recording. The system hardware part accepts the transcribed text from the application and is displayed upon execution of the multitude of note accessibility options.

Memory Enhancer application is a mobile memory assistant solution aimed specifically as a user-centered self-care intervention for dementia individual or independent living older adults who have self-determined as experiencing early-stage memory loss.

1.3 Testing Scope

This Testing is used to measure the overall quality of the developed Memory Enhancement application for its correctness, completeness, usability, and performance through functional and non-functional tests. The scope of the testing identifies what to be tested to ensure the performance implication of the system as per its requirements specification. The scope of this report therefore covered the execution of the required test suite against Memory Enhancement application by taking into consideration all the system requirements and as shown below.

Table 1: Requirements Traceability Matrix

Requirement #	Requirements
MEA001	The application shall be trained to recognize the user's voice.
MEA002	The application shall be Section 508 compliant such that an elderly person may use it without eyeglasses.
MEA003	The application shall recognize the user's unique voice and phrases then will automatically begin recording a memo.
MEA004	The application should ignore everything except what the user speaks.
MEA005	The application shall feature a turn-on button to start capturing user's speech.
MEA006	The application shall record transcribed text.
MEA007	The application shall replay transcribed text to user when the user requests with unique phrases.
MEA008	The application shall pair notes with subject of conversation.
MEA009	The application shall not save any recorded voice audio.
MEA010	The application shall allow user to edit and update text.
MEA011	The application shall have a flexible and functioning GUI (Graphical User Interface) that is user friendly.
MEA012	The application shall provide embedded training video guides to display the apps functionalities.
MEA013	The application shall integrate with IBM Watson Speech-to-Text for software

	transcription.
MEA014	The application shall run on the Android operating systems for mobile devices
MEA015	The application shall be built using the Dart programming language.
MEA016	The application's user interface (UI) will be built utilizing Flutter, an open-source UI toolkit.
MEA017	The application must encrypt the data at rest.
MEA018	The application shall retain speech to text recognition notes for 1 week in duration.
MEA019	The application shall provide the ability to search through the saved speech to text notes via text field and/or voice command.
MEA020	The application shall retrieve all results related to the search command.
MEA021	The application shall bypass asking everyone permission to record.
MEA022	The application shall provide a notification banner as a daily event presented to the user
MEA023	The application shall run on iOS for mobile devices.
MEA024	The application shall be scalable
MEA025	The application shall identify cost-license
MEA026	The application shall identity the level of effort to maintain
MEA027	The application shall have a customizable approach
MEA028	The application shall have a daily notification reminder banner

1.4 Intended Audience

The intention of this document is to act as a documented testing activities of memory enhancement application for stakeholders' reference. This is intending for the Design team, Developers, Project Manager, Business Analyst, Security analyst, Maintenance team and Testing team and end user for better understanding of the system performances. It is also intended for others that are undertaking similar project activities in the future.

1.5 Technical Project Stakeholders

A project is successful when it achieves its objectives and meets or exceeds the expectations of the stake-holders. Stakeholders defined in this document refers to individuals who have a vested interest and expertise to the design and development of this project. See Table 1 below for the list of stakeholders of the project.

Name	Role	Contact Info
Shawn Kelly	Project Manager	shawn_kelly123@yahoo.com
Chauntika Broggin	Developer	chauntika@gmail.com
Momodou Drammeh	Developer	momodou.drammeh1@gmail.com
Nicholas Ballo	Test Engineer	nbsockem@gmail.com
Christian Ahmed	Lead Developer	christian.ahmed@gmail.com
Ayodeji Famudehin	Business Analyst	aydot4life@yahoo.com
Mitchell Olshansky	Lead Tester	mitch.olshansky@protonmail.com

2. Overview of Test Results

2.1 Test Roles and Responsibilities

The roles and responsibilities of the testing team are displayed in the table below.

Table 2: Test Roles and Responsibilities

Team Member Name	Role	Responsibility
Mitch Olshansky	Test Engineer/Lead	Functional, End to End, Regression testing
Chauntika Broggin	Developer	End to End, Integration, Regression testing
Christian Ahmed	Developer	Integration, Unit testing
Momodou Drammeh	Developer	End to End, Unit testing
Nicholas Ballo	Test Engineer	Functional, Regression testing
Ayodeji Famudehin	Business Analyst	Functional, Unit testing.

2.2 Overall Assessment

Overwhelming majority of the tests performed on Memory Enhancer application passed. Out of 59 test cases, 5 bugs were reported and fixed. These were reproduced in multiple emulator environments. Observations were made during integration and end-to-end testing and all requirements from the stakeholders were satisfied. We will further discuss observations, bugs, and issues found in their respective sections.

2.3 Impact of Test Environment

The Memory Enhancement application was tested on multiple emulators of mobile phones running the Android or iOS operating systems. Emulation requires a large amount of processing power, so depending on the hardware being used for testing the application could be slow or unresponsive at times. High-end PCs can emulate the application on both Android and iOS without issue.

The following is a list of emulator models that have been used to test the application:

Nexus 6 - 1440 x 2560 / Android 11.0
Nexus 5X - 1080 x 1920 / Android 10.0
Pixel 3a - 1080 x 2220 / Android 11
Pixel 4 XL - 1440 x 3040 / Android 7.1.1
Pixel 2 - 1080 x 1920 / Android 8.1
iPhone 11 - 2532 x 1170 / iOS 14.0
iPhone 11 - 2532 x 1170 / iOS 13.0

The application was also tested on real mobile phones. All current reports indicate that the application operates without issue on any modern phone that was used during the testing process.

The following is a list of phone models that have been used to test the application:

Samsung Galaxy S9 – 2960 x 1440 / Android 10.0
Moto G Power – 1080 x 2300 / Android 10.0
Samsung S21 Ultra – 1440 x 3200 / Android 11.0
iPhone 12 – 2532 x 1170 / iOS 14.7

3. Test Summary

Table 3: Test Summary

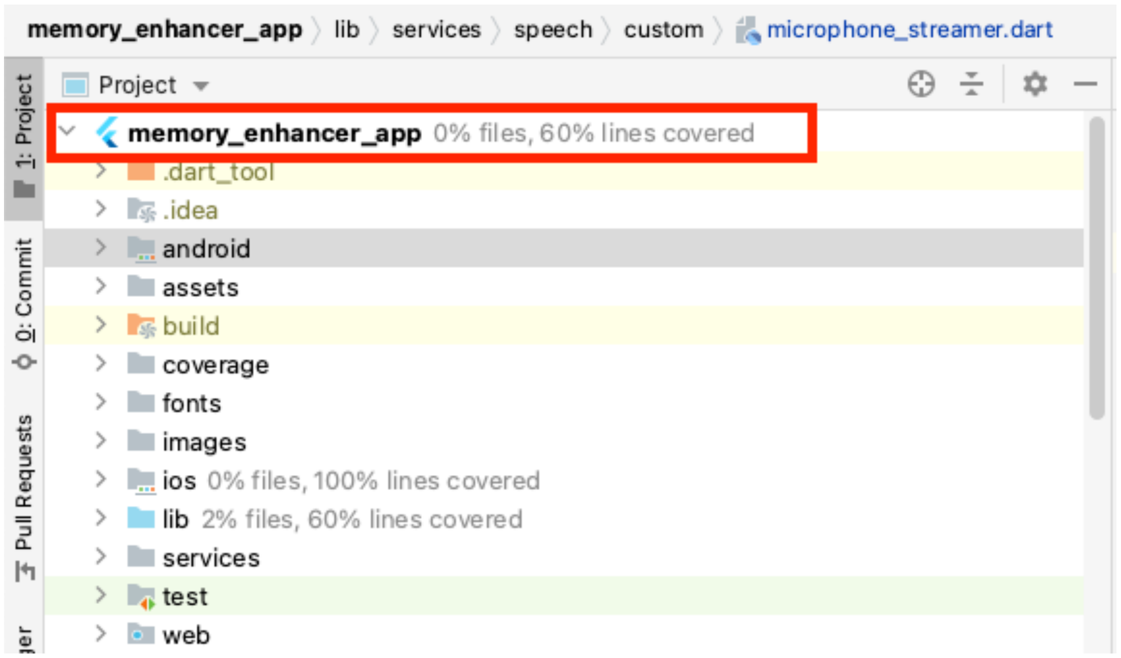
Tests Executed	Tests Passed	Tests Failed
59	59	0

We initially had 83 test cases, but after internal discussion, it was decided that we did not need as many overlapping test cases. A couple small bugs were discovered, but ultimately the tests passed. We executed collectively 59 test cases across a wide variety of environments from emulator to physical mobile devices; All with different screen resolutions and operating system versions. Our focus was on audio transcription, diarization process, feedback and answers, manage notes function, voice activation, data encryption and usability.

4. Facets of Testing

4.1 Unit Testing

Unit testing runs between code commits to the main branch in order to confirm integration of new code sections is working for individual features. The goal of unit testing is to an automated way to confirm all is working as expected with the latest updates and in their respective modular sector within the MVVM structure. 60% of all code lines have been unit tested.



memory_enhancer_app 〉 lib 〉 services 〉 speech 〉 custom 〉 microphone_streamer.dart

Project

- memory_enhancer_app 0% files, 60% lines covered
  - .dart_tool
  - .idea
  - android
  - assets
  - build
  - coverage
  - fonts
  - images
  - ios 0% files, 100% lines covered
  - lib 2% files, 60% lines covered
  - services
  - test
  - web

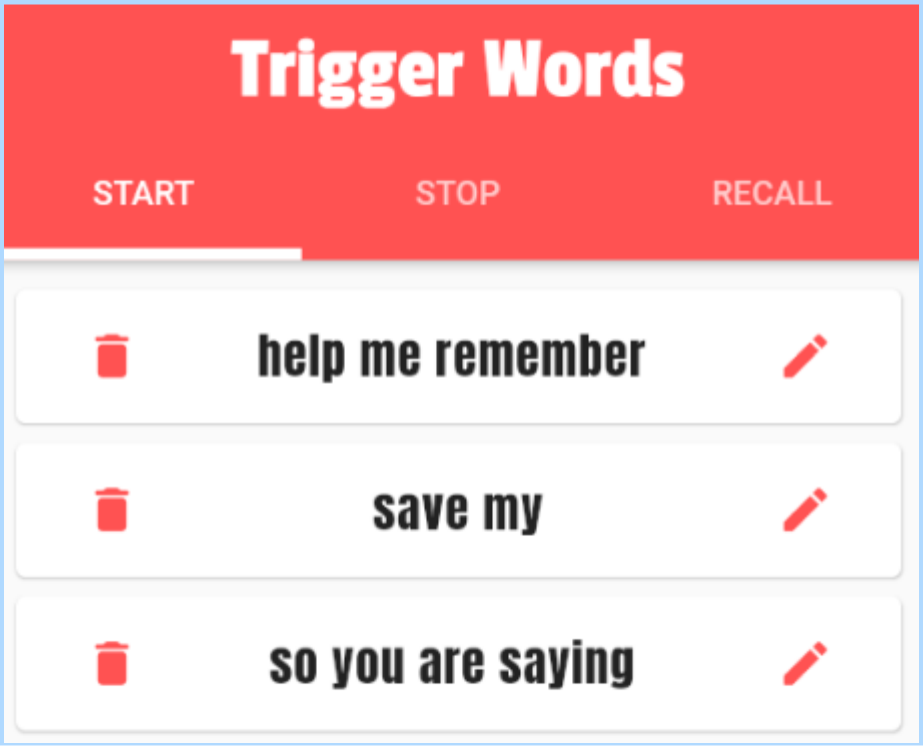
Figure 1 Memory Enhancement app's project displaying the use of unit testing

Some of the code lines not covered by unit testing are User Interface (UI) code lines.

4.2 Integration Testing

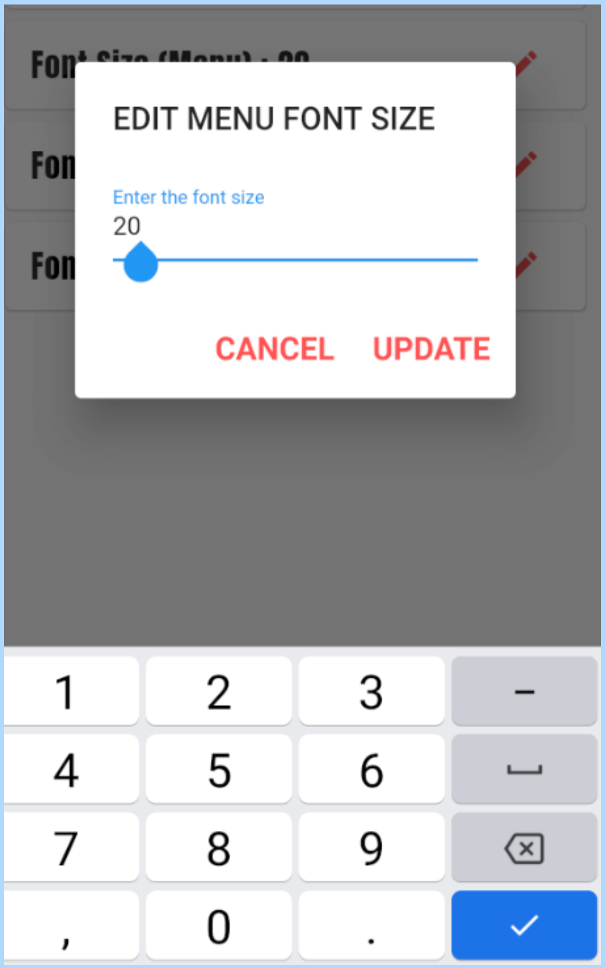
Test Case 1: Train Voice Recognition

Description:	**This test verifies that a user is able to train voice recognition within the application.**
Test Type:	User Acceptance
Requirement:	The application shall be trained to recognize the user's voice.
Req. #:	MEA001
Prerequisites:	
Steps:	1. User turns app on 2. User speaks trigger phrases/words 3. User repeats this process until application is trained 4. User logs off application
Expected Output:	Application recognizes certain words to trigger events based on functional scenarios.

Actual Output:	 Trigger Words START STOP RECALL help me remember save my so you are saying
Assumption:	Application has been created.

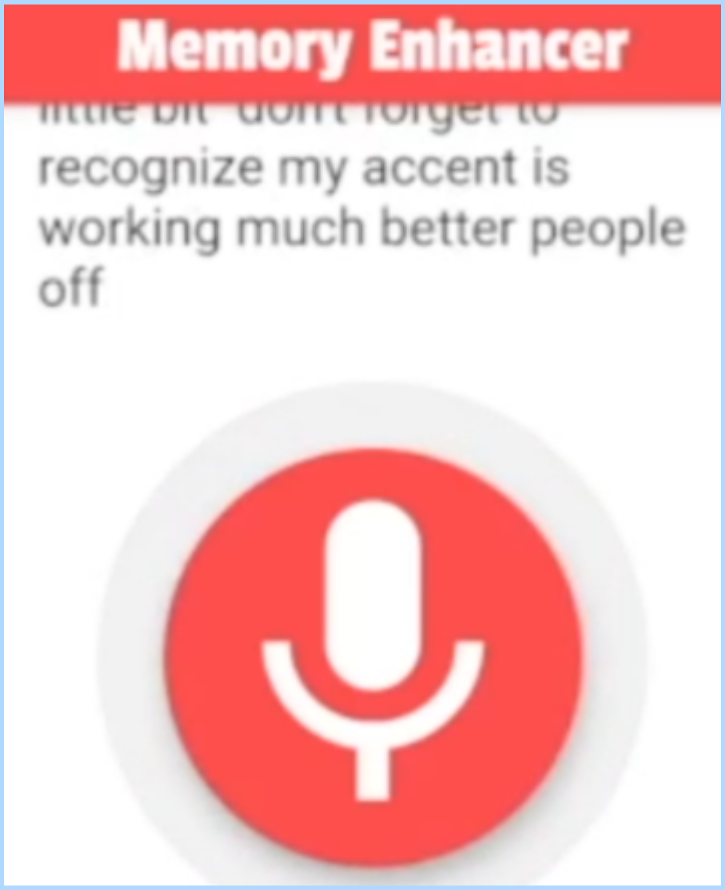
Test Case 2: 508 Compliancy

Description:	**This test verifies 508 compliancy in terms of font size, color contrast, etc. to support potential userbase.**
Test Type:	508
Requirement:	The application shall be section 508 compliant such that an elderly person may use it without eyeglasses.
Req. #:	MEA002
Prerequisites:	None
Steps:	1. User turns app on 2. User presses zoom in button 3. Application font and GUI increases in size 4. User presses zoom out button 5. Application font and GUI decreases in size
Expected Output:	GUI meets 508 standards through Zoom In/Out function and usability.

Actual Output:	 EDIT MENU FONT SIZE Enter the font size 20
Assumption:	GUI is functional for a sight-challenged user.

Test Case 3: Automatic Voice Detection

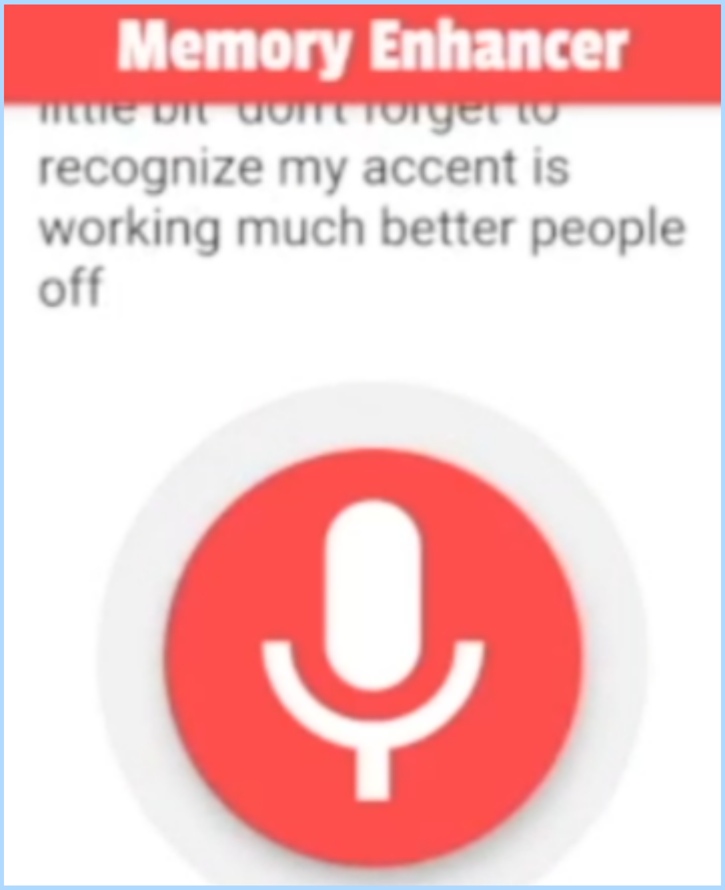
Description:	**This test verifies that the software will automatically begin transcribing the user's voice once it is recognized that they are speaking.**
Test Type:	Functional & User Acceptance
Requirement:	The application shall recognize the user's unique voice and phrases then will automatically begin recording a memo.
Req. #:	MEA003
Prerequisites:	The app has been trained to recognize the user's voice.
Steps:	1) The user turns app on. 2) The user begins speaking and mentions a phrase that the app has been trained to recognize.
Expected Output:	The application displays the user's words as text on screen.

Actual Output:	 Memory Enhancer little bit don't forget to recognize my accent is working much better people off
Assumption:	None

Test Case 4: Irrelevant Speech Avoidance

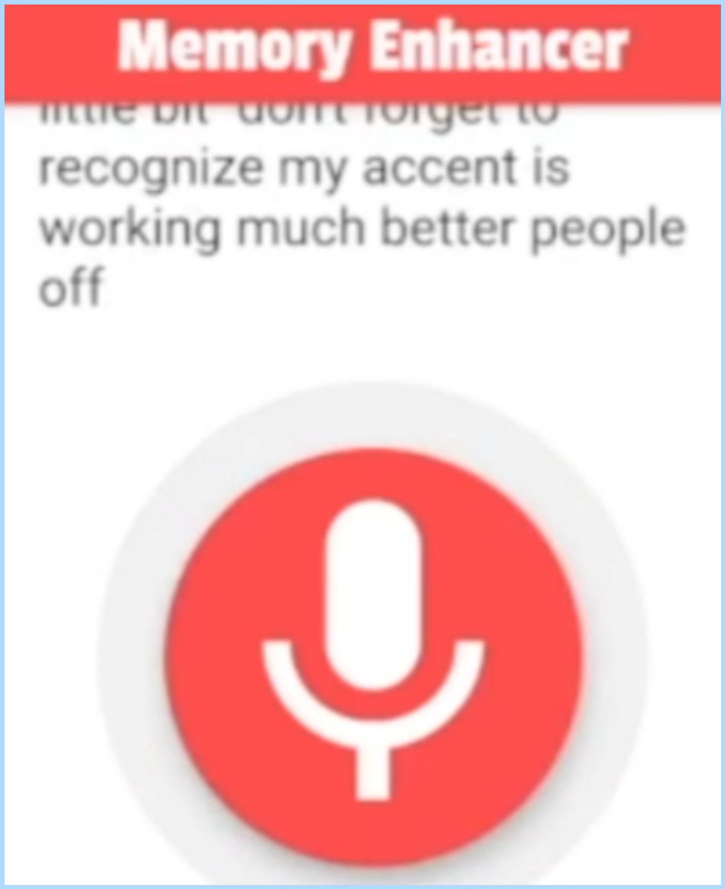
Description:	**This test verifies that irrelevant speech and noises will not be detected by the application.**
Test Type:	Functional & User Acceptance
Requirement:	The application should ignore everything except what the user speaks.
Req. #:	MEA004
Prerequisites:	The app has been trained to recognize the user's voice.
Steps:	1) The user turns app on. 2) The user either presses the record button or says a phrase that causes the app to begin recording. 3) Someone other than the user that the app has been trained for begins speaking.
Expected Output:	The app does not record or display the text of the non-user that is speaking.
Assumption:	None.

Test Case 5: Record Button Functionality

Description:	**This test verifies that speech is transcribed after the user has pressed the capture/record button.**
Test Type:	Functional & User Acceptance
Requirement:	The application shall feature a turn-on button to start capturing user's speech.
Req. #:	MEA005
Prerequisites:	The app has been trained to recognize the user's voice.
Steps:	1) The user opens the app 2) The user presses the capture/record button 3) The user begins speaking
Expected Output:	The app transcribes and displays the user's speech as text.
Actual Output:	 little bit don't forget to recognize my accent is working much better people off
Assumption:	None.

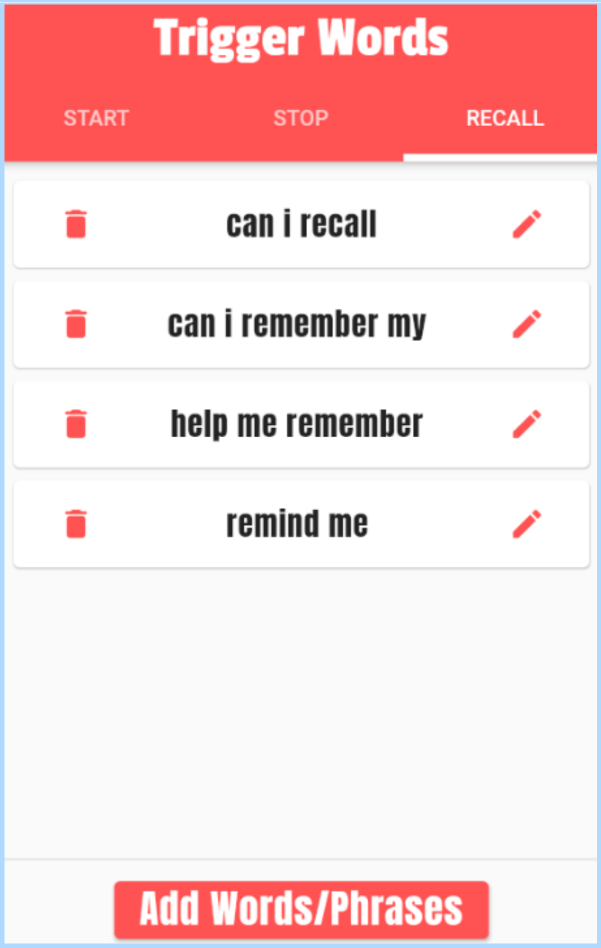
Test Case 6: Text Display

Description:	**This test verifies that text is displayed on screen that matches the user's speech.**
Test Type:	Functional & User Acceptance
Requirement:	The application shall record transcribed text.
Req. #:	MEA006
Prerequisites:	The app has been trained to recognize the user's voice.
Steps:	1) The user opens the app

	2) The user presses the capture/record button or says a phrase that causes the app to begin recording. 3) The user begins speaking.
Expected Output:	The app transcribes and displays the user's speech as text.
Actual Output:	 Memory Enhancer little bit don't forget to recognize my accent is working much better people off
Assumption:	None.

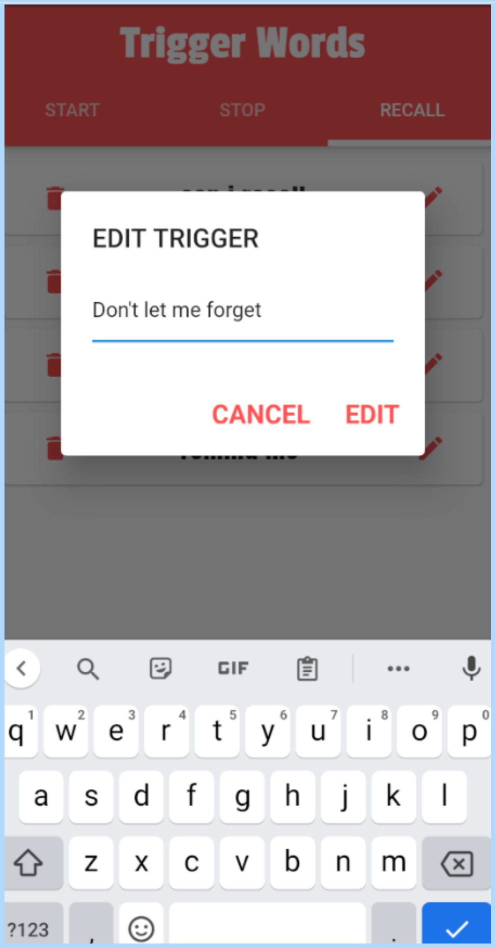
Test Case 7: Trigger Words – Add Words/Phrases

Description:	**This test verifies that text is displayed on screen that matches the user's speech.**
Test Type:	Functional & User Acceptance
Requirement:	The application shall record transcribed text.
Req. #:	MEA017
Prerequisites:	The application is able to store words and phrases
Steps:	1 Load application on emulator 2 Click on Settings button 3 Verify User lands on Settings page 4 Verify 'Settings' header in white text with red background 5 Click on and verify 'Trigger Words' subsection 6 User lands on Trigger Words subsection 7 Verify small red trashcan icons are present next to each trigger word/phrase 8 Verify small red pencil icons are present next to each trigger word/phrase 9 Verify 'Add Words/Phrases' red button with white text is at bottom of

	'Trigger Words' subsection 10 Click on "Add Words/Phrases" button 11 Type in phrase of your choice 12 Click Add button 13 Verify new phrase is added successfully to the Trigger Words subsection/page
Expected Output:	The app transcribes and displays the user's speech as text.
Actual Output:	 Trigger Words START STOP RECALL can i recall can i remember my help me remember remind me Add Words/Phrases
Assumption:	Notes Section of application has been completed.

Test Case 8: Trigger Words – Edit Words/Phrases

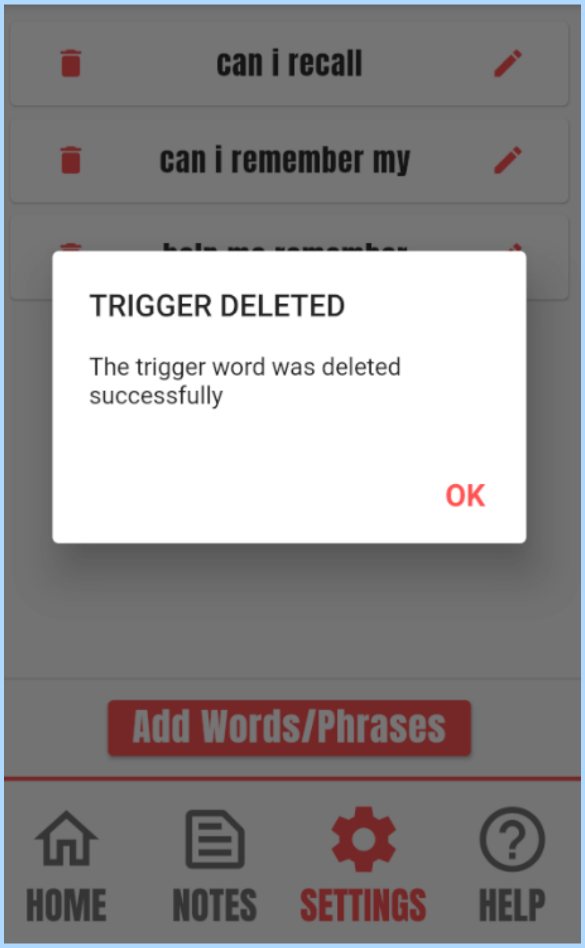
Description:	**This test verifies that text is displayed on screen that matches the user's speech.**
Test Type:	Functional & User Acceptance
Requirement:	The application shall record transcribed text.
Req. #:	MEA018
Prerequisites:	The application is able to store words and phrases
Steps:	1 Load application on emulator 2 Click on Settings button 3 Verify User lands on Settings page

	4 Verify ‘Settings’ header in white text with red background 5 Click on and verify ‘Trigger Words’ subsection 6 User lands on Trigger Words subsection 7 Verify small red trashcan icons are present next to each trigger word/phrase 8 Verify small red pencil icons are present next to each trigger word/phrase 9 Verify ‘Add Words/Phrases’ red button with white text is at bottom of ‘Trigger Words’ subsection 10 Click on red pencil next to “So you are saying” phrase 11 Verify external window generates with the respective phrase (So you are saying, in this case) 12 Verify red Edit button in white text generates 13 Change text to “Are you saying” 14 Click on Edit button 15 Verify phrase is changed in real time and reflected on Trigger Words subsection/page.
Expected Output:	The app transcribes and displays the user’s speech as text.
Actual Output:	
Assumption:	Notes Section of application has been completed.

Test Case 9: Trigger Words – Delete Words/Phrases

Description:	**This test verifies that text is displayed on screen that matches the user’s**

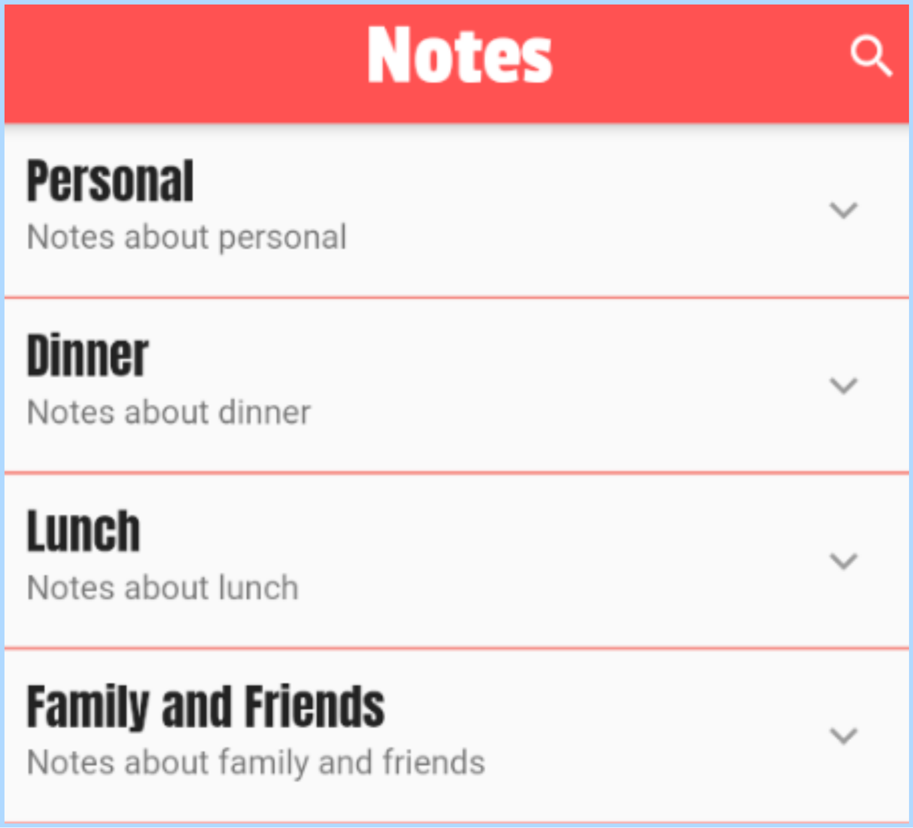
	speech.
Test Type:	Functional & User Acceptance
Requirement:	The application shall record transcribed text.
Req. #:	MEA019
Prerequisites:	The application is able to delete words and phrases
Steps:	1 Load application on emulator 14 Click on Settings button 15 Verify User lands on Settings page 16 Verify 'Settings' header in white text with red background 17 Click on and verify 'Trigger Words' subsection 18 User lands on Trigger Words subsection 19 Verify small red trashcan icons are present next to each trigger word/phrase 20 Verify small red pencil icons are present next to each trigger word/phrase 21 Verify 'Add Words/Phrases' red button with white text is at bottom of 'Trigger Words' subsection 22 Click on red trashcan next to "dinner" 23 Verify 'Confirm deletion' external window generates with red 'Delete' button in white text 24 Click on 'Delete' button 25 Verify "dinner" word/phrase is removed in real time from Trigger Words subsection
Expected Output:	The app transcribes and displays the user's speech as text.

Actual Output:	 can i recall can i remember my help me remember TRIGGER DELETED The trigger word was deleted successfully OK Add Words/Phrases HOME NOTES SETTINGS HELP
Assumption:	Notes Section of application has been completed.

Test Case 10: Message Replay

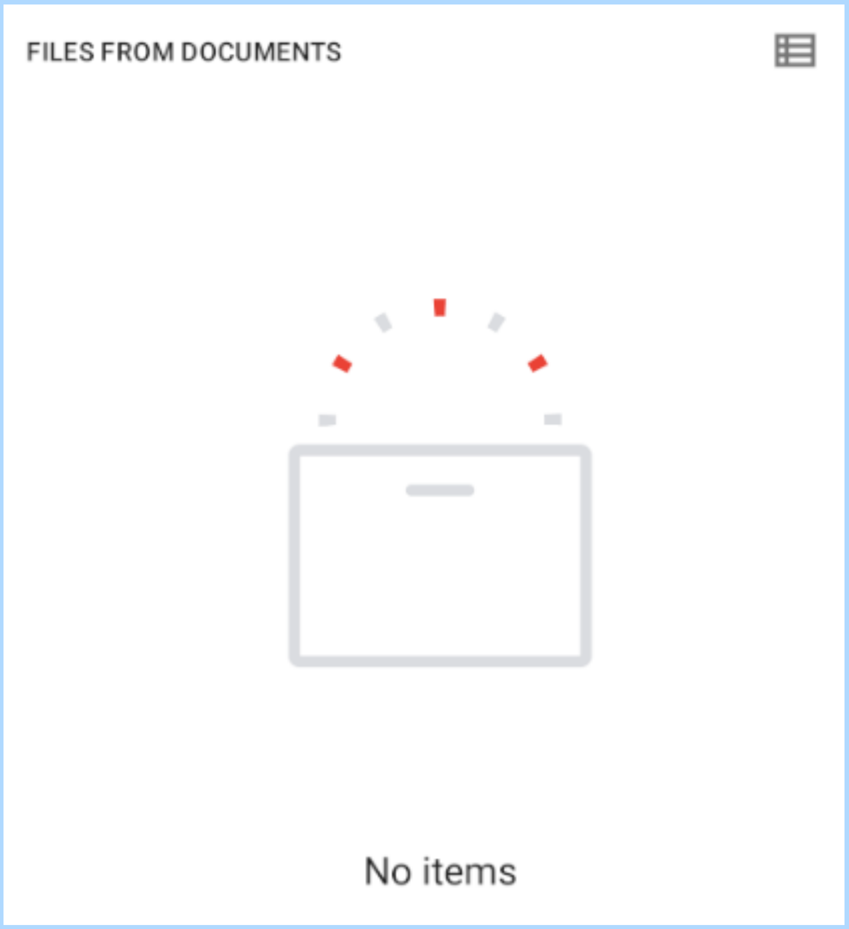
Description:	**This test verifies that the application is capable of replaying messages when upon the user's request.**
Test Type:	Functional & User Acceptance
Requirement:	The application shall replay transcribed text to user when the user requests with unique phrases.
Req. #:	MEA007
Prerequisites:	The user has previously recorded a message that is saved in the application.
Steps:	1) The user opens the app 2) The user selects a message that has been saved. 3) The user says a phrase that the app has been trained to recognize for message replay.
Expected Output:	The application replays the selected message using text-to-speech software.
Assumption:	The user's device is unmuted.

Test Case 11: Subject Recognition

Description:	**This test verifies that the application recognizes certain subjects when recording speech.**
Test Type:	Functional & User Acceptance
Requirement:	The application shall pair notes with subject of conversation.
Req. #:	MEA008
Prerequisites:	The app has been trained to recognize the user's voice.
Steps:	1) The user opens the app 2) The user presses the capture/record button or says a phrase that causes the app to begin recording. 3) The records multiple messages regarding the same subject
Expected Output:	The application groups the messages with the same subject together
Actual Output:	 Notes Personal Notes about personal Dinner Notes about dinner Lunch Notes about lunch Family and Friends Notes about family and friends
Assumption:	The application has been trained to recognize the subject

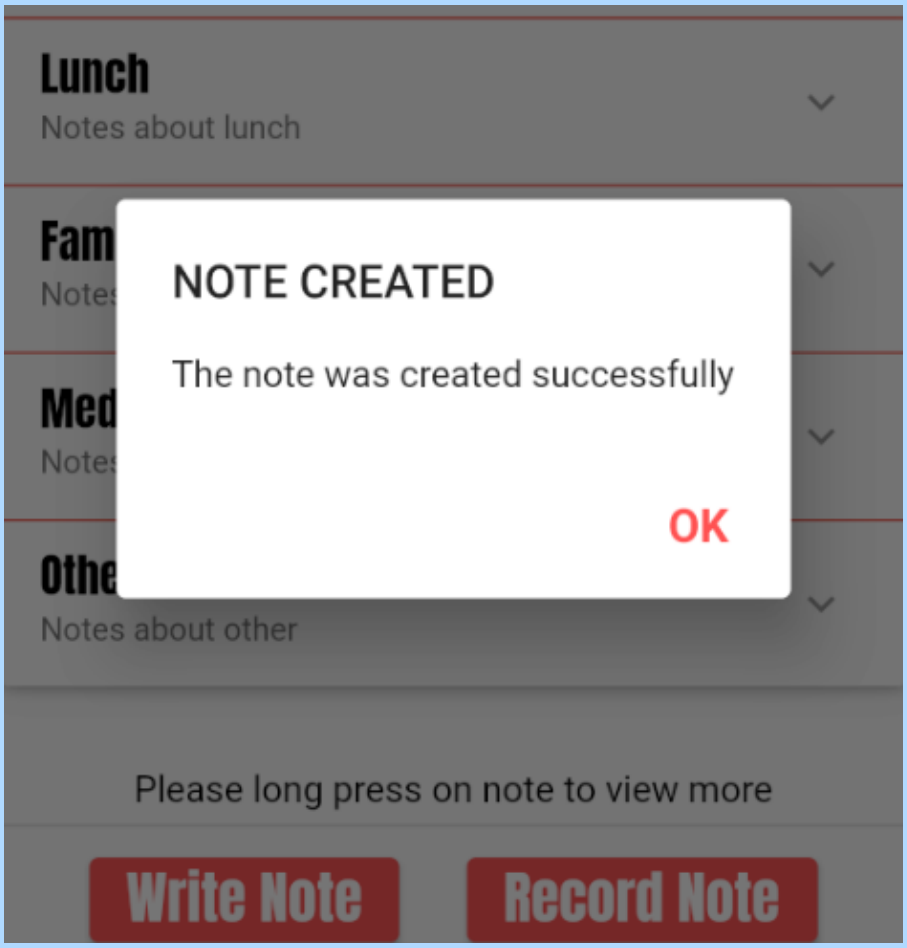
Test Case 12: Automatic Data Deletion

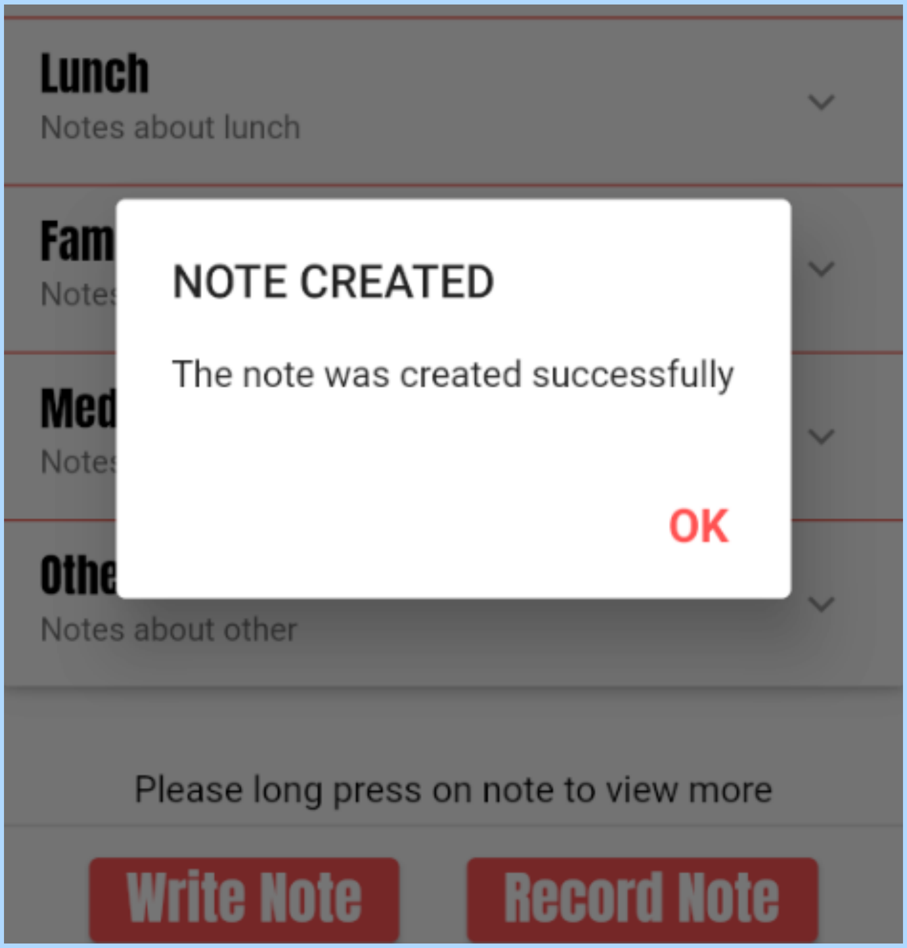
Description:	**This test verifies that no audio is stored on the user's device.**
Test Type:	Functional
Requirement:	The application shall not save any recorded voice audio.
Req. #:	MEA009
Prerequisites:	The app has been trained to recognize the user's voice.
Steps:	1) The user opens the app 2) The user presses the capture/record button or says a phrase that causes the

	app to begin recording 3) The user records a message 4) The user closes the app 5) The user view's the devices saved data
Expected Output:	The user does not find any saved audio from the application.
Actual Output:	 FILES FROM DOCUMENTS No items
Assumption:	The user's device contains space to theoretically save audio files.

Test Case 13: Text editing

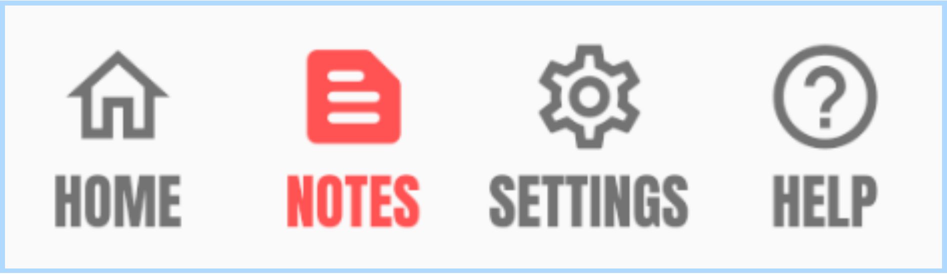
Description:	**This test verifies that the application allows the user to modify text that has been transcribed from speech.**
Test Type:	Functional & User Acceptance
Requirement:	The application shall allow user to edit and update text.
Req. #:	MEA010
Prerequisites:	The app has been trained to recognize the user's voice.
Steps:	1) The user opens the app

	2) The user presses the capture/record button or says a phrase that causes the app to begin recording 3) The user records a message 4) The user taps on the displayed message 5) An on-screen keyboard is displayed allowing the user to make modifications to the message 6) The user makes modifications and presses “Save”
Expected Output:	The modified message is saved and displayed in the application
Actual Output:	 NOTE CREATED The note was created successfully Please long press on note to view more
Assumption:	None



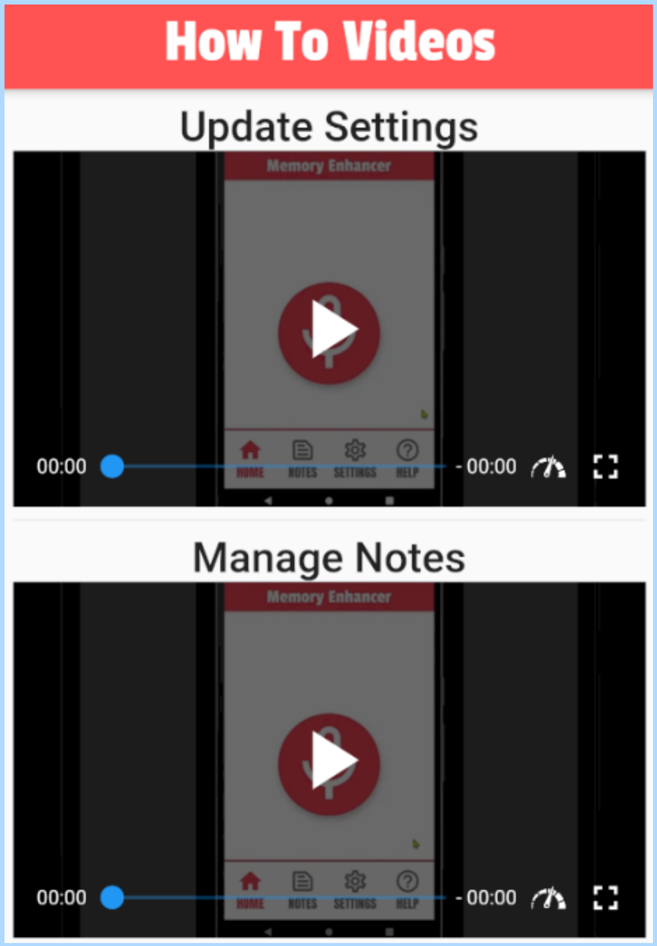
Test Case 14: Navigation

Description:	**This test verifies that the application is built intuitively so that users can learn to use it quickly.**
Test Type:	User Acceptance
Requirement:	The application shall have a flexible and functioning GUI that is user friendly.
Req. #:	MEA011
Prerequisites:	None.
Steps:	1) The user opens the app 2) The user presses on the various menu screens (help, settings, etc.)
Expected Output:	The application displays each menu as the user selects them.

Actual Output:	
Assumption:	None.

Test Case 15: Training Videos

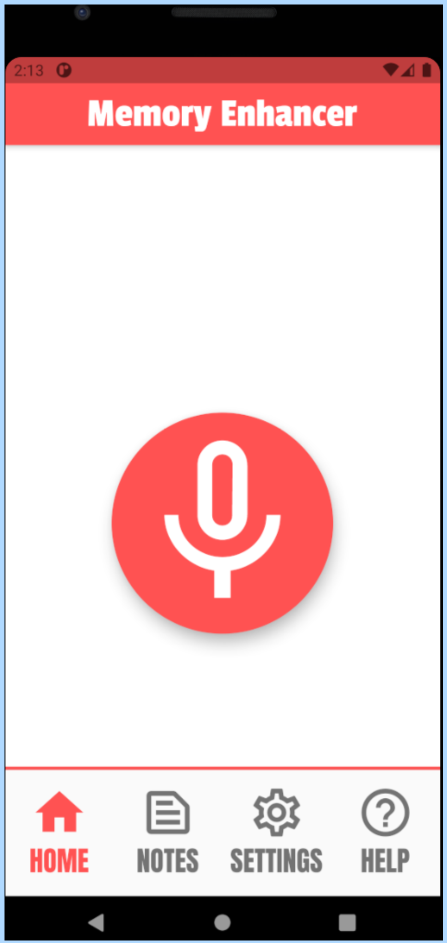
Description:	**This test verifies that the user can access training videos that explain how to operate the software.**
Test Type:	Functional & User Acceptance
Requirement:	The application shall provide embedded training video guides to display the apps functionalities.
Req. #:	MEA012
Prerequisites:	None
Steps:	1) The user opens the app 2) The taps the help menu icon 3) The user taps on a training video
Expected Output:	The application begins playing a video that explains how to operate the application.

Actual Output:	 How To Videos Update Settings Memory Enhancer 00:00 HOME NOTES SETTINGS HELP - 00:00 Manage Notes Memory Enhancer 00:00 HOME NOTES SETTINGS HELP - 00:00
Assumption:	None

Test Case 16: IBM Watson Speech-to-Text API Integration

Description:	**This test verifies that the software integrates with IBM's Watson Speech-to-Text software for speech transcription.**
Test Type:	Functional & User Acceptance
Requirement:	The application shall integrate with selected APIs.
Req. #:	MEA013
Prerequisites:	None.
Steps:	1) The application uses IBM Watson Speech-to-Text software when transcribing the user's speech
Expected Output:	Text is displayed on screen after being transcribed with Speech-to-Text.
Assumption:	None.

Test Case 17: Android Capability

Description:	**This test verifies that the application can run on the Android operating system.**
Test Type:	Functional
Requirement:	The application shall run on the Android operating systems for mobile devices.
Req. #:	MEA014
Prerequisites:	The user is using an Android device.
Steps:	1) The user opens the app 2) The user ensures that the app's various features are functional
Expected Output: **Actual Output:**	The app works correctly on the Android device.  2:13 Memory Enhancer HOME NOTES SETTINGS HELP
Assumption:	None.

Test Case 18: Built Using Dart Language

Description:	**This test verifies that the application has been coded in the Dart language.**
Test Type:	Functional
Requirement:	The application shall be built using the Dart programming language.
Req. #:	MEA015
Prerequisites:	None
Steps:	1) The application meets all requirements and is developed using the Dart language.
Expected Output:	None
Assumption:	None

Test Case 19: Built Using Flutter Framework

Description:	This test verifies that the application utilizing the Flutter framework.
Test Type:	Functional
Requirement:	The application's user interface (UI) will be built utilizing Flutter, an open-source UI toolkit.
Req. #:	MEA016
Prerequisites:	None
Steps:	1) The application meets all requirements and is developed using the Flutter framework.
Expected Output:	None
Assumption:	None

4.3 End to End Testing

In order to validate the entire application, the testing focus was on an end-to-end approach. Every feature and requirement tested in the most complete build available, against other features and requirements that overlap in functionality. We had a "no stone left unturned" approach. We tested for all dependencies including voice transcription, GUI navigation/interface, and testing on real devices in a variety of background settings where crowd chatter, television, or kitchen appliances can be heard. We wanted to make sure we exercised real world scenarios, supplying maximum test coverage.

4.3.1 Usability

Usability was handled from a 508-like perspective in order to target the key demographic of individuals who would find benefit in the application. We incorporated a Zoom in Zoom out function for individual with sight-related issues, as well as a layout with high level of contrast. We tested across several screen resolutions and operating systems' versions to improve usability in the greatest number of users possible. Memory Enhancer Application was loaded onto physical Android and iOS mobile devices as well and manually tested.

4.3.2 Performance Testing

Performance testing was done on several modern Android and iOS smartphones. All devices used were manufactured within the past five years and were running a popular variation of their respective operating systems.

Both practical visual tests and system diagnostic tests were performed. Visual tests included monitoring the application for stuttering or unresponsiveness. Diagnostic tests monitored the battery and RAM usage of the application. Test results indicated that the application has no trouble running efficiently on any modern device.