Short-Term Memory System (STeMS) Software Requirements Specification By AlphaSoft

<u>Team Members</u>

Awasthi, Aaditya

Blavat, Oleksiy

Bond, Matthew

Carter, Mackenzie

Mahbobi, Sayed shah

McAllister, Charlie

McCool, Max

McLaughlin, Taylor

Powers, Michael

Weaver, Daniel

Revision History

Date	Version	Description	Author
20 May 2023	1.0	Initial submission to Client	AlphaSoft
17 June 2023	2.0	Revisions to match TDD and updated PP	AlphaSoft
22 July 2023	3.0	Updated to match current state of project.	AlphaSoft
5 August 2023	4.0	Updated to match current state of project.	AlphaSoft

Table of Contents

Figures & Tables

1. Introduction

Natural Language Processing aims to combine the rules of human language with artificial intelligence and deep learning models, giving computers the ability to process large quantities of human language data. ChatGPT is one the leading generative AIs utilizing this technique in order to create written language responses to user given prompts. AlphaSoft is seeking to explore the different application areas for ChatGPT that would benefit those struggling with short-term memory loss via the Short-Term Memory System (STeMS.) Combined with the natural language processing capabilities of ChatGPT, the STeMS will give users recollection assisting tools for in person conversations.

This document is the Software Requirements Specification (SRS) for the Talker mobile application development effort.

1.1. Purpose

The purpose of this SRS is to document the primary requirements for the proposed application. The use cases provided describe the ways users may interact with STeMS in conjunction with any additional requirements needed for proper functionality. The AlphaSoft Development team will use this document as guidance for the front-end development of the application. The duties for the front-end development effort will include all UI/UX design efforts, user input validation, error handling, and the creation of reusable widgets for optimized speed and scalability. All testing efforts will cover only the front-end portions of the application, back-end functionality will be tested separated by the appropriate development team. This SRS is being prepared in conjunction with the back-end development team, which will document their requirements separately. Subsequent sections of this SRS will also document any potential constraints or limitations for STeMS.

This SRS is a living document that will be maintained throughout the entirety of the development process. All revisions will result in an updated document version number. Revision dates, version numbers, and descriptions will be documented in the revision history table provided.

1.2. Scope

The scope of this SRS covers the user interface and the user interaction aspects of STeMS. The application will allow users to create and manage live conversation recordings. Users will have the ability to select how recordings will be processed. This SRS also outlines an additional browser extension that will allow the application to use processed recordings to populate webpage forms. This SRS covers only the front-end features for the application. All back-end implementation is outside the responsibility of the AlphaSoft Development team and will be outlined in a separate SRS document.

This document will not address any design decisions from an aesthetic standpoint and focuses only on the functionality of the application. Any additional features for the proposed applications found necessary for proper functionality will be added to this SRS in later versions.

1.3. Project Documents

The SRS document is part of a set of documents created to aid in developing the STeMS application and to provide artifacts with vital information for the application's ongoing support and operation throughout its full life cycle.

The following documentation will be included in the entire documentation package for this project.

Document	Version	Date
Project Plan (PP)	4.0	5 August 2023
Software Requirements Specification (SRS)	4.0	5 August 2023
Technical Design Document (TDD)	3.0	5 August 2023
Software Test Plan (STP)	3.0	5 August 2023
Programmer Guide (PG)	2.0	5 August 2023

Document	Version	Date
Deployment and Operations Guide (DOG)	2.0	5 August 2023
Software Test Report (STR)	1.0	5 August 2023
User Guide (UG)	1.0	5 August 2023
Traceability Matrix (TM)	1.0	5 August 2023

Table 1: Project Documents

1.4. Acronyms Definitions, and Abbreviations

Throughout this SRS, a variety of terms and acronyms specific to the proposed application are used. For clarity, their definitions are provided below.

Term	**Definition**
AI	Artificial Intelligence
API	Application Programming Interface, or a way that difference applications interact with each other
App	A program that is included on the App User's mobile device
BESie	Back End Service
Browser Extension	An extension to either the Chrome or Safari browser, which works with the App to provide specific functionality
Conversation	A recording the App User made and saved within the app
Mobile Device	A smart phone, tablet, or some other portable computer with either the iOS or Android operating system
PG	Programmer's Guide
PP	Project Plan
Deployment and Operations Guide	A document that explains how to deploy the program
STeMS	Short-Term Memory System
STP	Software Test Plan
STT	Speech to Text
System	The complete STeMS system, which includes the App, the Brower Extension, and the Browser Extension Service
SRS	Software Requirements Specification
TDD	Technical Design Document
TR	Test Report
UI/UX	User Interface / User Experience
UG	User Guide

Table 2: Acronyms, Definitions and Abbreviations

1.5. References

Assadullah, M., & Wilson, R. (2023, May 13). Kickoff Meeting. Retrieved from https://umgcdev361.sharepoint.com/sites/SWEN670Summer2023/_layouts/15/stream.aspx?id=%2Fsites%2FSWEN670Summer2023%2FShared%20Documents%2FGeneral%2FRecordings%2FKickoff%2D20230513%5F170521%2DMeeting%20Recording%2Emp4

Brown, M. (2020, March 24). *Complete Project Guide.* Retrieved from SWEN 670 Capstone Project Guide: https://learn.umgc.edu/d2l/le/content/769776/viewContent/30624571/View

Dart. (n.d.). *Dart Overview*. Retrieved from Dart: https://dart.dev/overview

OpenAI. (2022, November 30). *Introducing ChatGPT*. Retrieved from OpenAI: https://openai.com/blog/chatgpt

OpenAI. (n.d.). *API Reference*. Retrieved from OpenAI: https://platform.openai.com/docs/api-reference

1.6. Overview

The remainder of this SRS will consist of two primary sections, Overall Description and Specific Requirements. The Overall Description will provide a high-level overview of the requirements for the proposed system. The Specific Requirements will provide a more detailed definition of each use case. A follow-on section will provide any supplementary information considered necessary for the understanding of this document.

2. Overall Description

The primary function of the app is to record conversations and make them available to be acted upon by AI. The major pieces of that workflow are the recording and management of conversations, the transcribing of those conversations to text (STT), and the capability of ChatGPT to act upon that information in the process of providing further functionality.

2.1. Guiding Factors

This product is user driven and derives its purpose and requirements from the needs of the users in their most common settings. The use cases specify the memory impaired users directly, and the application could easily assist the hearing impaired as well with its speech to text functionality. As these cases revolve around the use of ChatGPT and AI in adaptive functioning for the differently abled, the focus shall be prioritizing the features for those users to deliver the most helpful product possible.

The requirements of this software are led by a few general rules. The first is that the software must be intuitive and simple enough to operate by someone without above average computer skills. STeMS should take into account that users will be comprised of widely varying ages and technical competencies, so the app should aim to simplify all functions with that in mind. Since the app will primarily be used to capture life events as they are happening, STeMS will also minimize delays and make the app as easy to utilize on the fly as possible. Finally, while the intuitive nature of the app is important, continuing ease of use is paramount to making the app adoptable long term. Organization, searchability, and data management all will help users retain key information and make it accessible. By focusing on these three elements, the app will maximize its effectiveness for new and continuing users of all backgrounds.

Because the product will be assisting those with short term memory issues and potentially a large group of older users, the UI should be plain and easy to comprehend at first glance. A very basic home page with easily recognizable function buttons should make options obvious, and buttons should be large enough to be seen and easily accessed. Where text is required, sizing options should be considered.

Many apps that aim to assist users in their daily life often must contend with the “startup hurdle,” which is the time it takes to get the app to function. In cases where conversations are ongoing, it’s crucial to minimize startup time and allow users to open and engage the app as quickly as possible. Like opening a camera app in time to catch “the moment,” so too must STeMS be ready to capture crucial details in conversations, whether they be at a doctor's office, in a work meeting, or even with friends and family. Every effort should be made to minimize delays from app start to the “ready to record” state.

Once these crucial details are recorded, users must be given the tools to maximize the information gathered. After recording, users shall be able to label conversations. After saving, users should be able to search by title to find information from past conversations. In order to manage storage space, options should exist to delete recordings.

Future iterations of the app may include further options to add details or elements to the recording and the ability to search by those. Timed auto-delete for unused recordings, and an ability to “lock” a recording could also be in future versions.

2.2. Typical Users

Edna: Elderly and with severe memory issues, she also has eyesight problems. Has never used a computer and was given her phone by her grandson. She most often uses it when visiting the doctor, or to fill out applications based on previous conversations.

The Lawyer: Uses the program with consenting clients to maintain a searchable record of conversations. Tagged by client and case, it is used as a crucial reference tool.

Mike: Hearing impaired man in his mid-20’s. He’s very computer savvy but has trouble in short spontaneous conversations. Uses the program mostly when details are crucial, like technical conversations at work. Will use the program to save himself time filling out forms on the web.

Debbie: Hearing impaired waiter at small family restaurant. While she knows the regulars, she could use some help with the new patrons. Uses the recorder in her pocket as a backup when taking orders.

2.3. Use-Case Model Survey

2.3.1. Actors

- App User – A person that is using the system on a mobile device.
- Browser User – A person that is using a browser to surf the web. The browser may have the Browser Extension installed.
- Back End Service (BESie) - A web service that acts as a passthrough between the App and the Browser Extension.
- Whisper API – A web-based API that does STT conversions.
- ChatGPT API – A web-based API for an advanced language model AI.

2.3.2. Use Cases

Use Case	Description
Make Recording	High-level overview of the use cases contained in the New Recording Use Cases section.
Edit Recording Metadata	Overview of the process of editing the title of a conversation.
Search Recordings	The process of searching for specific recordings.
Delete Recording	Overview of the process of deleting a recording.
Sort Recordings	Overview of the process of sorting recordings.
Process Recording	High-level overview of the use cases contained in the Process Recording Use Cases section
Guided Tour	Overview of the tour given to the user to explain how to use the App.
Start Recording	The process of starting a recording.
Pause Recording	The process of pausing a recording.
Resume Recording	The process of resuming a recording.
Stop Recording	The process of stopping a recording.
Save Recording	The process of saving a recording.
Convert Speech to Text	The process of converting a recording to text, to ensure that the ChatGPT API can be used.
Select Recording	Overview of the process of selecting a recording to manipulate.
Select Process	Overview of the process of selecting which feature to use with a conversation.
Package Request	The process of gathering the text for ChatGPT.
Process Results	The process of retrieving a response from ChatGPT and deciding what to do with the results.
Generate Code	BESie generates a code to determine which Browser Extension request goes to which App instance.
Enter Code	A code is entered to connect to the correct App instance,
Receive Extension Request	The App determines if an incoming request should be performed.

Use Case	Description
Extension Response	The process of the App responding to the browser extension's request for a conversation.
Request to Populate Webpage	A request is made to the browser extension to populate a form on a webpage based on a conversation.
Populate Webpage	A form on a webpage is populated based on a selected conversation

2.4. Diagrams

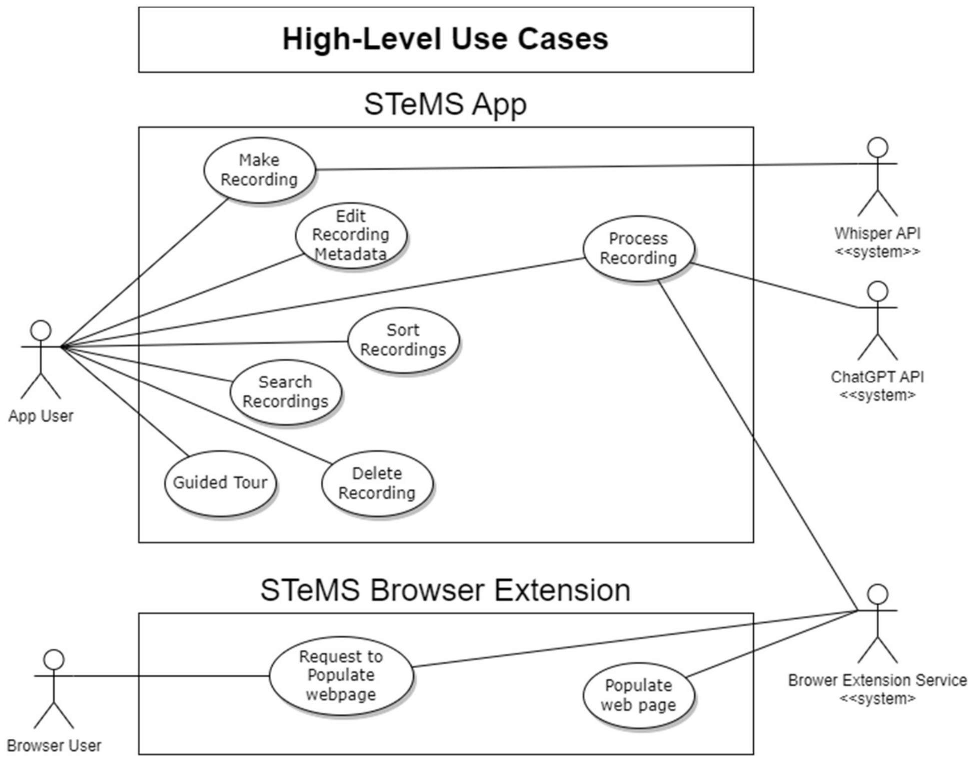


Figure 3: High-Level Use Cases

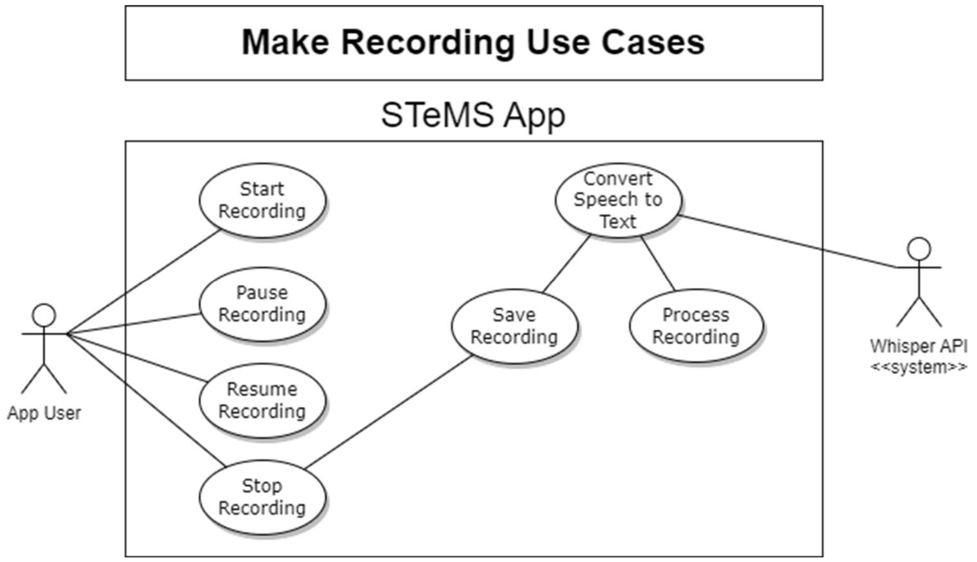


Figure 4: Make Recording Use Cases

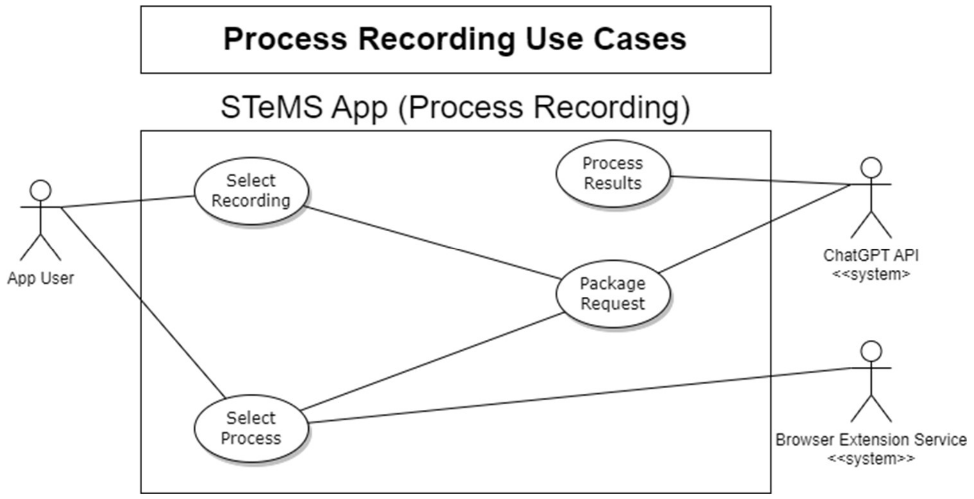


Figure 5: Process Recording Use Cases

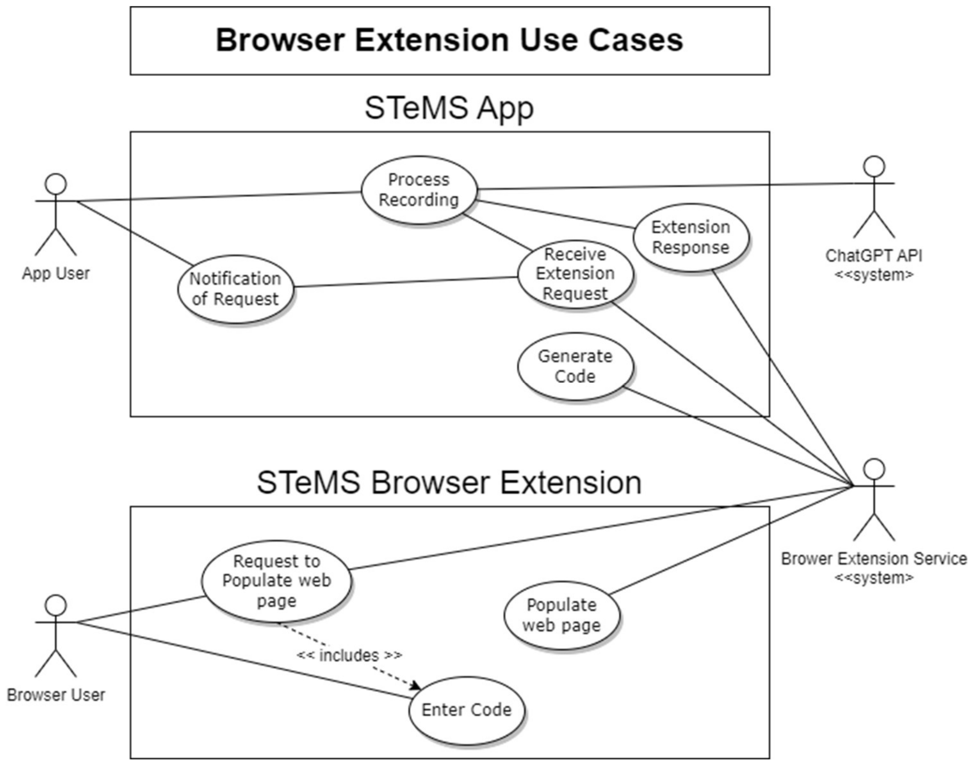


Figure 6: Browser Extension Use Cases

2.5. Assumptions and Dependencies

Development shall proceed assuming that the phone has a functioning microphone, and space capacity to store recordings. The initial tool will be developed within the constraints of the available technologies, with conversation length being an initial example of expected limitations due to available hardware and software capabilities. Development is assuming users will have either the Android or iOS mobile operating systems, as the app is being developed for compatibility with those. It is assumed users can add and operate the browser extension into their Chrome or Safari browsers.

The app is dependent on an internet connection for the STT and ChatGPT processing to work, but the recording, labeling, and storing functions should operate normally with or without it. STT does not have to happen at the moment of recording but can be triggered when the connection is available for it, or on command.

The browser extension shall be supported on Chrome, Firefox, Edge, and Safari. Because it uses Web API standards, it might work on other browsers, but AlphaSoft will only test on the four mentioned ones.

3. Specific Requirements

3.1. Global Rules

- All error messages will have user friendly text stating what the problem is and what the user can do about it.
- Any time an error message is shown, the App User will be returned to the last screen they were on. That screen will be in the same state as when they started the process that got the error.
- Users shall be required to confirm any decisions that would delete an existing recording, including pressing the back button throughout the recording process.

3.2. Use-Case Reports

3.2.1. High-Level Use-Cases

3.2.1.1. Make Recording

Summary: The user wants to make a new recording. These are detailed below in the referenced use cases.

Priority: High

Actors: App User

Preconditions: The App User has the app installed and open to the Conversation List Screen (4.1.1)

Triggers: The App User clicks on the "Start Recording" button.

Basic course of events (main scenario):

Step	System
1	The App User does the Start Recording use case.
2	The App User does the Stop Recording use case.
3	The App User does the Save Recording use case.

Alternate courses of events (alternate scenarios)

Step	System
A.2	The App User does the Pause Recording use case.
A.3	The App User does the Resume Recording use case.

Post-conditions: A new recording is created.

3.2.1.2. Edit Recording Metadata

Summary: A user wants to edit the information of a recording.

Priority: Low

Actors: App User

Preconditions: The App User must have a conversation saved and be on the Conversation Details Screen (4.1.4).

Triggers: The App User taps on a conversation's title.

Basic course of events (main scenario):

Step	System
1	The App automatically highlights the title.
2	The App User types in a new title.
3	The App User clicks the "Accept" button on their mobile device's keyboard.
4	The App saves the new title.

Alternate courses of events (alternate scenarios)

Step	System
A.2	The App User taps the Back icon.
A.3	The App redirects the user back to the Conversations List Screen (4.1.1)
B.2	The App User doesn't type anything.
B.3	The App keeps the same conversation title.

Post-conditions: The App saves the conversation with a new title.

3.2.1.3. Search Recordings

Summary: A user wants to look up a recording.

Priority: Medium

Actors: App User

Preconditions: The App User has already installed the app and has a conversation saved.

Triggers: The App User is on the Conversations List Screen (4.1.1).

Basic course of events (main scenario):

Step	**System**
1	The App User types something in the search bar.
2	The App presents a list of autocompleted conversation titles to the App User, based on what they type in.
3	The App User selects a word that the App suggests.
4	The App searches all saved conversations and presents the user with a list of conversations to click on based on the words typed by the App User.

Alternate courses of events (alternate scenarios)

Step	**System**
A.2	The App cannot find any conversation titles that match what the App User is typing and doesn't present any options to the App User.
A.3	The App User continues typing out a word or deletes some text.
B.4	The App returns no results and presents a "No conversations found" message where the conversations would normally appear (4.1.2).
C.2	The App takes too long to generate a list.
C.3	The App will continue to let the App User type text unhindered and will either catch up or not if the App User is moving faster than the App.

Post-conditions: The App User is presented with a list of conversations based on search results.

3.2.1.4. Delete Recording

Summary: A user wants to delete a conversation.

Priority: Medium

Actors: App User

Preconditions: The App User has performed the Save Recording use case.

Triggers: The App User taps on a conversation.

Basic course of events (main scenario):

Step	**System**
1	The App User clicks on the trashcan icon.
2	The App presents the user with a pop-up message that says, “Are you sure you want to remove this conversation permanently?” with the given options of “OK” or “Cancel” (4.1.7)
3	The App User clicks “OK.”
4	The App deletes the conversation.

Alternate courses of events (alternate scenarios)

Step	**System**
A.1	The App User taps the Back icon.
A.2	The App redirects the user back to the Conversations List Screen (4.1.1).
B.3	The App User taps “Cancel.”
B.4	The message box goes away, the conversation is not deleted.

Post-conditions: The conversation is deleted.

3.2.1.5. Sort Recordings

Summary: A user wants to sort their recordings.

Priority: Low

Actors: App User

Preconditions:

- The App User has already installed the app and has a few recordings saved.
- The App user is on the Conversations List Screen (Figure 4.1.1).

Triggers: The App User clicks the "Sort" icon.

Basic course of events (main scenario):

Step	**System**
1	The App presents the user with a few sorting options such as sorting by date, title, and duration (Figure 4.1.3).
2	The App User taps on a sorting option.
3	The App closes the list of sorting options.
4	The App sorts the list of conversations based on the sorting option chosen by the App User.

Alternate courses of events (alternate scenarios)

Step	**System**
A.2	The App User doesn't tap on a sorting option.
A.3	The App doesn't sort the conversations.
B.2	The App User taps anywhere on the screen that isn't the sorting list.
B.3	The App closes the list of sorting options.
B.4	The App does not change the current sorting behavior.

Post-conditions: The App sorts the list of conversations.

3.2.1.6. Process Recording

Summary: The App User wants to process a conversation. This is broken down in the Process Recording Use Cases section below.

Priority: High

Actors: App User, App

Preconditions: The App is installed.

Triggers: The App User is on the Conversation Details Screen (Figure 4.1.4).

Basic course of events (main scenario):

Step	System
1	The App User does the Select Recording use case.
2	The App User does the Select Process use case.
3	The App does the Package Request use case.
4	The App does the Process Results use case.

Alternate courses of events (alternate scenarios)

Step	System
A.2	The App User changes their mind and does not want to process this combination.
A.3	The App User may repeat either the Select Process or Select Recording use cases.
A.4	The App deselects any prior selected item of the same type (process or recording).
B.2	The App User wants to cancel.
B.3	The App User may either select on any blank space to deselect everything or tap the same process or recording again to deselect them.

Post-conditions: The conversation has been processed and results handled.

3.2.1.7. Guided Tour

Summary: The user gets a tutorial on how to use the App.

Priority: High/Medium/Low

Actors: App User

Preconditions: The App is installed.

Triggers:

- The App is opened for the first time
- The App User clicks on the "Guided Tour" button on the Information Screen (Figure 4.1.11).

Basic course of events (main scenario):

Step	**System**
1	The App displays a preset slideshow (Figure 4.1.12) showing the various screens with explanatory blurbs for the various features.
2	The App User can proceed forward or backward by clicking the left or right arrows respectively.
3	The App User reaches the final slide of the tour.
4	The App changes the right arrow to a "Done" button.
5	The App User clicks Done.
6	The App ends the guided tour and returns to the Conversation Screen.

Alternate courses of events (alternate scenarios)

Step	**System**
A.2	The App User clicks the system "Back" button at any point throughout the tour.
A.3	The App ends the guided tour and returns to Conversation Screen.

Post-conditions: The App User is on the Conversation Screen.

3.2.2. New Recording Use Cases

3.2.2.1. Start Recording

Summary: The user wants to start a new recording.

Priority: High

Actors: App User

Preconditions: The App is installed and open to the Conversation List Screen (Figure 4.1.1).

Triggers: The App User clicks on the Record button.

Basic course of events (main scenario):

Step	**System**
1	The App takes the App User to the Recording - Started Screen (Figure 4.1.8).
2	The App starts the recording.

Alternate courses of events (alternate scenarios)

Step	**System**
A.1	The App cannot start the recording.
A.2	The App displays an error message.

Post-conditions: The App starts the recording.

3.2.2.2. Pause Recording

Summary: The user wants to pause the recording.

Priority: Medium

Actors: App User

Preconditions: The App User performs the Start Recording use case.

Triggers: The App User clicks the “Pause” icon.

Basic course of events (main scenario):

Step	**System**
1	The App suspends the recording process without ending the recording completely.
2	The App changes to the Recording - Paused Screen (Figure 4.1.9).
3	The App waits for the Resume Recording use case or the Stop Recording use case to be performed.

Alternate courses of events (alternate scenarios)

Step	**System**
A.1	The App fails to pause the recording.
A.2	The App displays an error message.
B.3	The App User hits the “Back” button on their phone.
B.4	The App continues to record.
B.5	The App displays a pop-up “Do you want to stop the recording?” with Stop and Cancel options. See Recording – Stopped Screen (Figure 4.1.10).
B.6.a	The App User clicks Stop, which starts the Stop Recording use case.
B.6.b	The App User clicks Cancel.
B.6.c	The App cancels the Back button operation and continues as if it was never pressed.

Post-conditions: The recording has been paused.

3.2.2.3. Resume Recording

Summary: The user wants to resume a paused recording.

Priority: Medium

Actors: App User

Preconditions:

- The App User performs the Start Recording use case.
- The App User performs the Pause Recording use case.

Triggers: The App User clicks the "Resume" button.

Basic course of events (main scenario):

Step	System
1	The App changes to the Recording - Started Screen (Figure 4.1.8).
2	The App continues the recording, using the same stream/file.
3	The App waits for the Pause Recording use case or the Stop Recording use case to be performed.

Alternate courses of events (alternate scenarios)

Step	System
A.1	The App fails to resume the paused recording.
A.2	The App displays an error message.

Post-conditions: The recording has been resumed.

3.2.2.4. Stop Recording

Summary: The user wants to stop a recording.

Priority: High

Actors: App User

Preconditions: The App User performs the Start Recording use case.

Triggers:

- The App is recording.
- The App User taps the Stop button.

Basic course of events (main scenario):

Step	**System**
1	The App stops recording.
2	The App saves the recording locally.
3	The App auto-assigns a title to the conversation.
4	The App saves the conversation.
5	The App sends the App User to the Conversation Details Screen (Figure 4.1.4).
6	The App performs the Convert Speech to Text use case.
7	The App performs the Package Request use case using this conversation and the Summary Transmogrifier.
8	The App performs the Process Results use case for the previous Package Request.
7	The App performs the Package Request use case using this conversation and the Reminders Transmogrifier.
8	The App performs the Process Results use case for the previous Package Request.

Alternate courses of events (alternate scenarios)

Step	**System**
A.1	The App cannot stop the recording.
A.2	The App displays an error message.
B.2	The App fails to save the recording.
B.3	The App displays an error message.
C.3	The App fails to generate a title.
C.4	The App uses a hard-coded default title for the conversation.

Post-conditions: The recording is stopped.

3.2.2.5. Convert Speech to Text

Summary: The App needs to convert the recording to text so that the ChatGPT API can be used as well as to support other functionality.

Priority: High

Actors: App, Whisper API, App User

Preconditions: There a saved recording.

Triggers:

- The Stop Recording use case has successfully saved the conversation.
- The App User initiates a process to do something with a conversation and the text is missing.

Basic course of events (main scenario):

Step	**System**
1	The App sends the conversation’s audio file to the Whisper API.
2	The Whisper API returns the text of the conversation.
3	The App saves the text with the conversation’s data.

Alternate courses of events (alternate scenarios)

Step	**System**
A.1	The App cannot find/access/load into memory the audio file.
A.2	The App displays an error to the App User.
B.1	The App cannot connect to the Whisper API.
B.2	The App displays an error to the App User.
C.2	The Whisper API gets an error. It might or might not return the error to the App.
C.3	The App displays an error to the App User.
D.2	The Whisper API takes too long.
D.3	The App displays an error to the App User.
E.2	The Whisper API returns no text.
E.3	The App displays an error to the App User.

Post-conditions: The App keeps the conversation’s text in memory for further handling.

3.2.3. Process Recording Use Cases

3.2.3.1. Select Recording

Summary: A user needs to select a recording that they want to manipulate.

Priority: High

Actors: App User, App

Preconditions: The App is installed.

Triggers: The App User is on the Conversation List Screen (Figure 4.1.1).

Basic course of events (main scenario):

Step	**System**
1	The App User may or may not do the Search Recordings use case.
2	The App User taps the conversation they want to work with.
3	The App sends the App User to the Conservation Details Screen (4.1.4).

Alternate courses of events (alternate scenarios)

Step	**System**
A.1	There are no saved conversations.
A.2	The App displays in the Conversation list area the message “No Conversations Found.”
A.3	The App provides a way for the App User to start recording right away (the Start Recording use case), which is optional.

Post-conditions: The App notes the selected conversation for further processing.

3.2.3.2. Select Process

Summary: A user needs to select which process they want to run a conversation through.

Priority: High

Actors: App User, App

Preconditions: The App is installed.

Triggers:

- The App User is on the Conversation Details Screen and decides to do something with it (Basic Scenario).
- The App receives a request to process a conversation (see Receive Extension Request use case) (Alternate Scenario A).

Basic course of events (main scenario):

Step	System
1	The App User selects the process they want by tapping it.
2	The App changes the color of the process's background.
3	The App displays "Transmogrifying…" in the process's results area.

Alternate courses of events (alternate scenarios)

Step	System
A.1	The App receives an internal request to process a conversation.
A.2	The App tries to match the request with a valid process.
A.3	The App returns the matched process to the request.
B.A.2	The App cannot find a match.
B.A.3	The App returns a "no match found" response.

Post-conditions: The App notes the selected process for further handling.

3.2.3.3. Package Request

Summary: The App needs to gather everything for use by the ChatGPT API.

Priority: High

Actors: App, ChatGPT API, App User

Preconditions:

- Use case "Select Process" has been completed successfully.
- Use case "Convert Speech to Text" has been completed successfully.

Triggers: All preconditions are satisfied.

Basic course of events (main scenario):

Step	System
1	The App retrieves the conversation's text.
2	The App retrieves the selected process.
3	The App determines the prompt(s) to send to ChatGPT API.
4	The App sends the text to ChatGPT API in the proper order.

Alternate courses of events (alternate scenarios)

Step	System
A.1	The App cannot retrieve the conversation's text, or the text is empty.
A.2	The App displays an error to the App User.
B.2	The App cannot retrieve the selected process.
B.3	The App displays an error to the App User.
C.4	The App cannot connect to the ChatGPT API.
C.5	The App displays an error to the App User.
D.4	The App fails to send all text to ChatGPT API.
D.5	The App displays an error to the App User.
E.4	The ChatGPT API gets an error. It might or might not return the error to the App.
E.5	The App displays an error to the App User.
F.4	The ChatGPT API takes too long.
F.5	The App displays an error to the App User.
G.4	The ChatGPT API returns no text.
G.5	The App displays an error to the App User.

Post-conditions: The ChatGPT API returns a final response.

3.2.3.4. Process Results

Summary: The App must process the result returned by the ChatGPT API.

Priority: High

Actors: App, ChatGPT API, App User

Preconditions: The Package Request use case completed successfully.

Triggers: The Package Request use case completed successfully.

Basic course of events (main scenario):

Step	**System**
1	The App receives the final response from ChatGPT API.
2	The App, based on the selected process, determines what to do with this response.
3.a	If the selected process requires displaying text to the App User, then display the results in the Conversation Details results area.
3.b	If the selected process requires sending data back to BESie, then do the Extension Response use case.
3.c	If the selected process requires setting a reminder, then add this reminder to the list.

Alternate courses of events (alternate scenarios)

Step	**System**
A.3	If anything goes wrong, then the App displays an error to the App User.

Post-conditions:

- 3.a: The results area displays with the correct text.
- 3.b: The App displays a pop-up message that it completed the task.
- 3.c: The results area displays with the correct text.

3.2.4. Browser Extension Use Cases

3.2.4.1. Generate Code

Summary: BESie needs a way to know which Browser Extension request goes to which App instance.

Priority: Medium

Actors: App, BESie

Preconditions: The App is installed but has never been run.

Triggers: The App User starts the App for the first time.

Basic course of events (main scenario):

Step	System
1	The App will request a code from BESie. It will send identifying information with this request.
2	BESie will generate a unique code and store it in a permanent location along with the identifying information.
3	BESie will send the code back to the App.
4	The App will store it in a permanent location.

Alternate courses of events (alternate scenarios)

Step	System
A.4	The App cannot save/access permanent storage.
A.5	The App displays an error to the App User.
B.1	The App cannot connect to BESie.
B.2	The App displays an error to the App User.
C.2	BESie gets an error. It might or might not return the error to the App.
C.3	The App displays an error to the App User.
D.2	BESie takes too long.
D.3	The App displays an error to the App User.
E.2	BESie gets an error while processing the request.
E.3	BESie returns a failure message.
E.4	The App displays an error to the App User.

Post-conditions:

- If no errors, then the code is stored in a permanent location in the App.
- If no errors, then the code and identifying information are stored in a permanent location on BESie.
- If errors that the App is aware of, then the App will exit after reporting the error.
- If errors that BESie is aware of, then BESie will not store the code or the identifying information

3.2.4.2. Enter Code

Summary: The Browser Extension needs to know which App instance to connect to.

Priority: Medium

Actors: Browser User, Browser Extension

Preconditions: The Browser Extension is installed.

Triggers: The Browser Extension is building the request to send to BESie.

Basic course of events (main scenario):

Step	**System**
1	The Browser Extension asks the Browser User for the code.
2	The Browser User enters the code found in the App and clicks Fill Form.
3	The Browser Extension continues to build the request.

Alternate courses of events (alternate scenarios)

Step	**System**
A.2	The Browser User never enters a code.
A.3	The Browser Extension waits on the Browser User.
B.2	The Browser User enters an invalid character (if only numbers are allowed, then they are entering a non-numeric character, etc.).
B.3	The Browser Extension does not allow invalid characters to be typed, so the typed character will not appear on the screen.

Post-conditions: The Browser Extension has the data it needs to continue building the request.

3.2.4.3. Receive Extension Request

Summary: The App must receive and process requests from the Browser Extension.

Priority: Medium

Actors: App, BESie

Preconditions: The Request to Populate Webpage use case has successfully completed.

Triggers: BESie sends the request to the App.

Basic course of events (main scenario):

Step	**System**
1	The App receives the request.
2	The App does the Notification of Request use case. The result was Proceed.
3	The App determines which process needs to be performed based on the request.

Alternate courses of events (alternate scenarios)

Step	**System**
A.2	The result of the Notification of Request use case was Cancel.
A.3	The App stops processing the request.
A.4	The App does the Extension Response use case with a user canceled message.

Post-conditions: The App does the Process Recording use case.

3.2.4.4. Notification of Request

Summary: The App needs to confirm that an incoming request should be performed.

Priority: Medium

Actors: App User, App

Preconditions: The App has received a request from BESie.

Triggers: The App starts processing the request from BESie (the Receive Extension Request).

Basic course of events (main scenario):

Step	System
1	The App creates a phone notification stating that a request to process a conversation was received, and it will have 2 options: Proceed and Cancel.
2	The App User selects Proceed.
3	The App opens and displays the Conversation List Screen.

Alternate courses of events (alternate scenarios)

Step	System
A.2	The App User selects Cancel.
A.3	The App sends a Cancel response back to BESie.
B.2	The App User does not select either option when the notification appears.
B.3	The App User opens the App.
B.4	The App will display the same notification message if it has not timed out yet.

Post-conditions: The result of this use case will be passed back to the Receive Extension Request use case.

3.2.4.5. Extension Response

Summary: The App needs to respond to the Browser Extension request for data.

Priority: Medium

Actors: App, BESie

Preconditions: The Receive Extension Request use case has started.

Triggers:

- The Process Recording use case has finished successfully.
- The Receive Extension Request use case, alternate scenario A has completed.

Basic course of events (main scenario):

Step	**System**
1	The App receives the data from the triggering use case.
2	The App builds the correct response to the request.
3	The App sends the response to BESie.

Alternate courses of events (alternate scenarios)

Step	**System**
A.2	The App gets an error building the response.
A.3	The App displays an error to the App User.
B.3	The App cannot connect to BESie.
B.4	The App displays an error to the App User.
C.3	BESie gets an error processing the response. It might or might not return the error to the App.
C.4	The App displays an error to the App User.

Post-conditions: BESie has the data and can now forward it to the Browser Extension.

3.2.4.6. Request to Populate Webpage

Summary: A user requests the browser extension to populate a form in a webpage.

Priority: Medium

Actors: Browser User, Browser Extension

Preconditions: The Browser Extension is installed.

Triggers: The Browser User is on a webpage and wants to populate the form from a conversation.

Basic course of events (main scenario):

Step	**System**
1	The Browser User selects the “Populate from Conversation” option from within the Browser Extension.
2	The Browser Extension does the Enter Code use case.
3	The Browser Extension parses the DOM to get needed information.
4	The Browser Extension sends the request to BESie.

Alternate courses of events (alternate scenarios)

Step	**System**
A.3	The Browser Extension gets an error during parsing.
A.4	The Browser Extension displays an error to the Browser User.
B.4	The Browser Extension cannot connect to BESie.
B.5	The Browser Extension displays an error to the Browser User.
C.4	BESie gets an error. It might or might not return the error to the Browser Extension.
C.5	The Browser Extension displays an error to the Browser User.
D.4	BESie takes too long to send a response.
D.5	The Browser Extension displays an error to the Browser User.

Post-conditions: The form is populated with answers.

3.2.4.7. Populate Webpage

Summary: The system populates the form on the webpage based on the selected conversation.

Priority: Medium

Actors: Browser User, BESie, Browser Extension

Preconditions: The Process Extension Request use case has completed successfully.

Triggers: BESie sends the results to the Browser Extension.

Basic course of events (main scenario):

Step	**System**
1	The Browser Extension receives the results from BESie.
2	The Browser Extension confirms that the URL and/or DOM is the same as used in the Request to Populate Webpage use case.
3	The Browser Extension recursively searches the DOM for a match.
4	The Browser Extension updates attributes/nodes in the DOM as needed to populate the matching result into the DOM.

Alternate courses of events (alternate scenarios):

Step	**System**
A.1	The Browser Extension never receives the results from BESie.
A.2	The Browser Extension takes too long.
A.3	The Browser Extension displays an error to the Browser User.
B.2	The Browser Extension catches that the URL and/or DOM is not the same.
B.3	The Browser Extension displays an error to the Browser User.
C.3	The Browser Extension cannot find a match.
C.4	The Browser Extension skips that item and continues looking for the next item.
D.4	The Browser Extension gets an error while doing the update.
D.5	The Browser Extension displays an error to the Browser User.

Post-conditions: The Browser User can see the webpage has been fully updated based on the results sent back.

3.3.Supplementary Requirements

3.3.1. Security

- The mobile application will use a bearer token when sending sensitive information to OpenAI's APIs (example: transcribed user conversations, audio files).
- The token will not be stored in plain text in the application. Instead, we will be using the flutter_dotenv library to load environment variables and keys necessary for the application's core functions.
- We only need one set of tokens to talk to OpenAI's Whisper API for audio transcription and ChatGPT/completion API endpoint. This means there is only one vendor dependency.
- All conversation data and transcripts will be stored on user's mobile device. Once the user uninstalls the app, the audio conversations and their transcripts will be deleted and cannot be restored.

3.3.2. Interface constraints

- The application will run on modern iOS and Android mobile devices that are still supported by the manufacturer.
- The app will only run in vertical mode (portrait).
- That app will use responsive design to support mobile devices of all sizes.
- The UI shall be simple and easy for the user to interact with by using clear and concise language and by avoiding clutter.
- The interface should incorporate highly legible text and high contrast buttons to make navigation easy for those with poor eyesight.
- Ensure that the application displays useful and informative error messages to the user when errors occur.

3.3.3. Performance constraints

- To avoid long wait times, audio splitting, and poor user experience due to transcription mistakes, the voice recording will be capped at 25MBs in size which is roughly equivalent to 25-30 minutes of audio recording.
- The WebSocket connection to the pass-through service responsible for handling communication back and forth between the browser extension and the app will time out within 60 seconds of a user receiving a notification to fill out a form. This constraint reduces the impact on phone's battery usage and user's cell phone data usage
- All recorded conversations will be automatically transcribed once the recording is finished. The app will store a plain-text version of the conversation on the user's phone. This will help us reduce the number of live API calls we need to make for the form filler feature and any other future features. It will also support the feature that will allow the search to search all conversations' text.

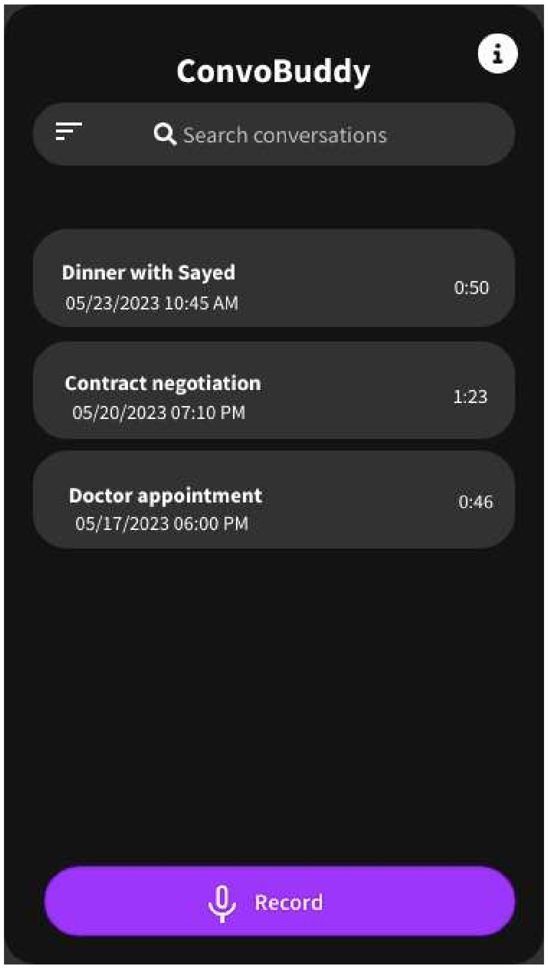
3.3.4. Accessibility

Accessibility is an integral aspect of our software development process. We are committed to ensuring that our software is accessible and usable by individuals of all abilities, including those with disabilities. Our team adheres to established accessibility standards, such as the Web Content Accessibility Guidelines (WCAG) and employs best practices to provide an inclusive user experience. We prioritize features like keyboard navigation, screen reader compatibility, color contrast, and clear visual indicators to enhance accessibility. By designing and developing our software with accessibility in mind, we aim to provide equal access and usability to all users, promoting inclusivity and diversity in our digital solutions.

4. Supporting Information

4.1. Use Case Screen Shots

4.1.1. Conversation List Screen



Dinner with Sayed
05/23/2023 10:45 AM
0:50

Contract negotiation
05/20/2023 07:10 PM
1:23

Doctor appointment
05/17/2023 06:00 PM
0:46

Figure 7: Conversation List Screen

4.1.2. Conversation List- Search Not Found Screen

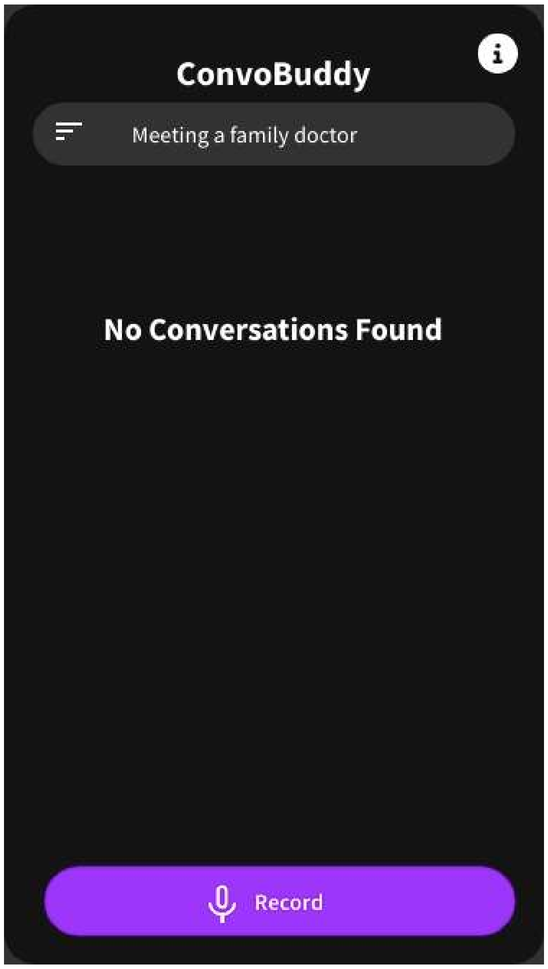


Figure 8: Conversation List - Search Not Found Screen

4.1.3. Conversation List– Sort Screen

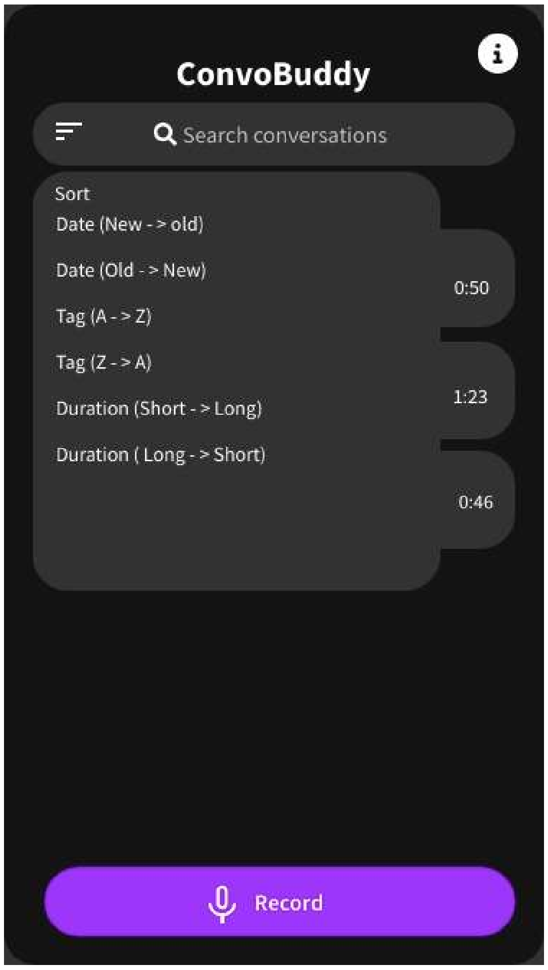
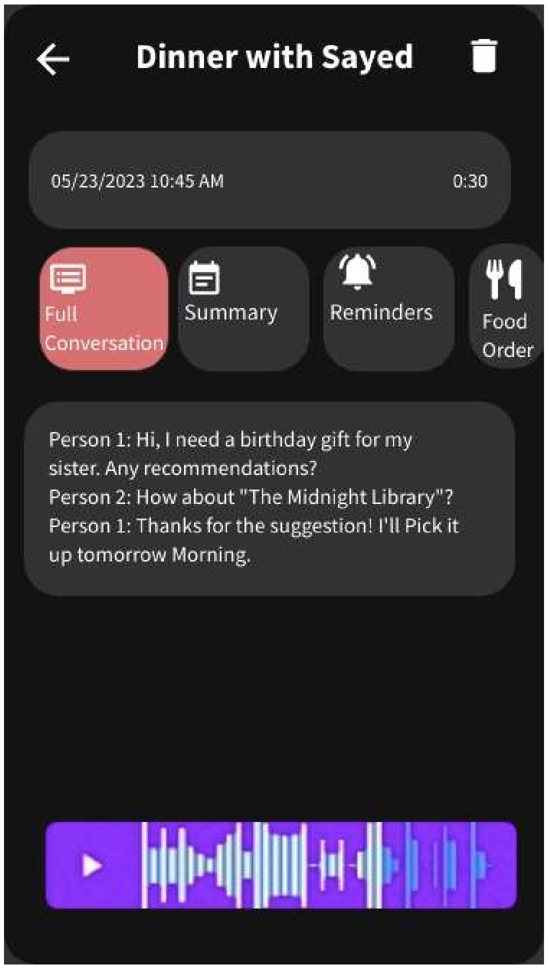


Figure 9: Conversation List - Sort Screen

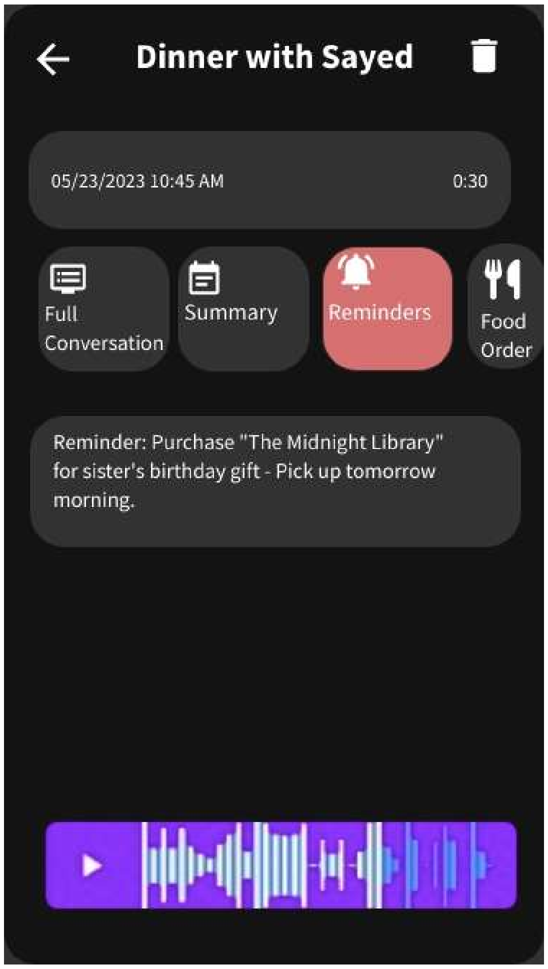
4.1.4. Conversation Details Screen



Person 1: Hi, I need a birthday gift for my sister. Any recommendations?
Person 2: How about "The Midnight Library"?
Person 1: Thanks for the suggestion! I'll Pick it up tomorrow Morning.

Figure 10: Conversation Details Screen

4.1.5. Conversation Details – Reminders screen



Reminder: Purchase "The Midnight Library" for sister's birthday gift - Pick up tomorrow morning.

Figure 11: Conversation Details - Reminders Screen

4.1.6. Conversation Details – Missing Result Screen

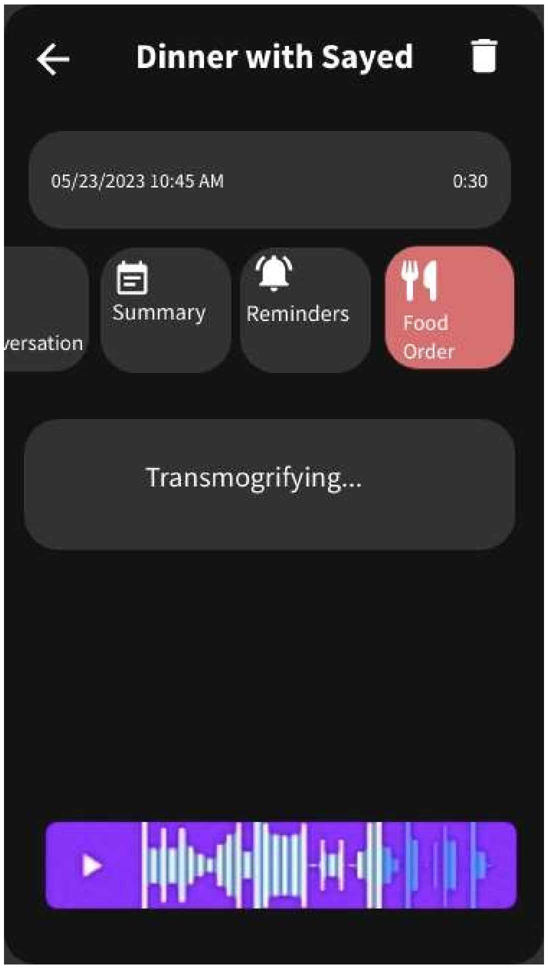


Figure 12: Conversation Details - Missing Result Screen

4.1.7. Conversation Details - Delete Screen

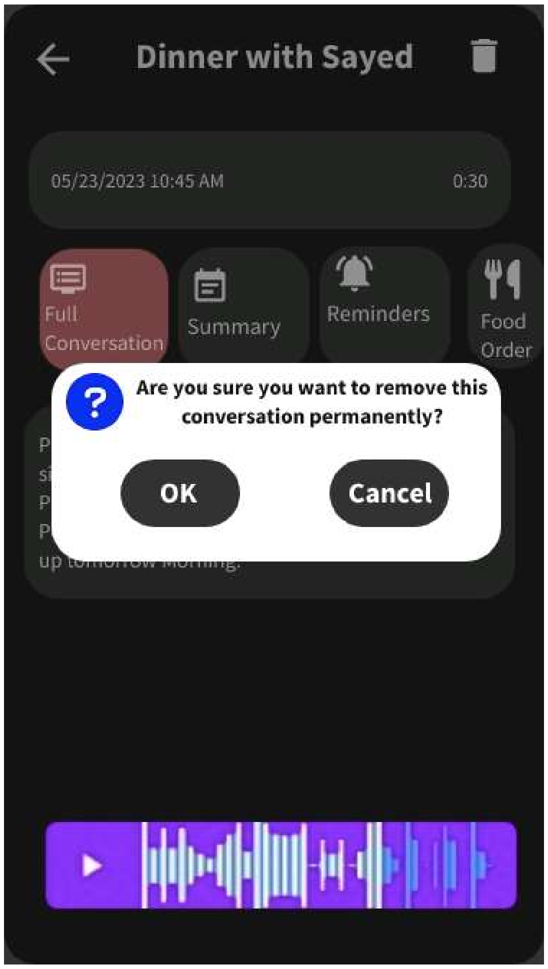


Figure 13: Conversation Details - Delete Screen

4.1.8. Recording - Started Screen

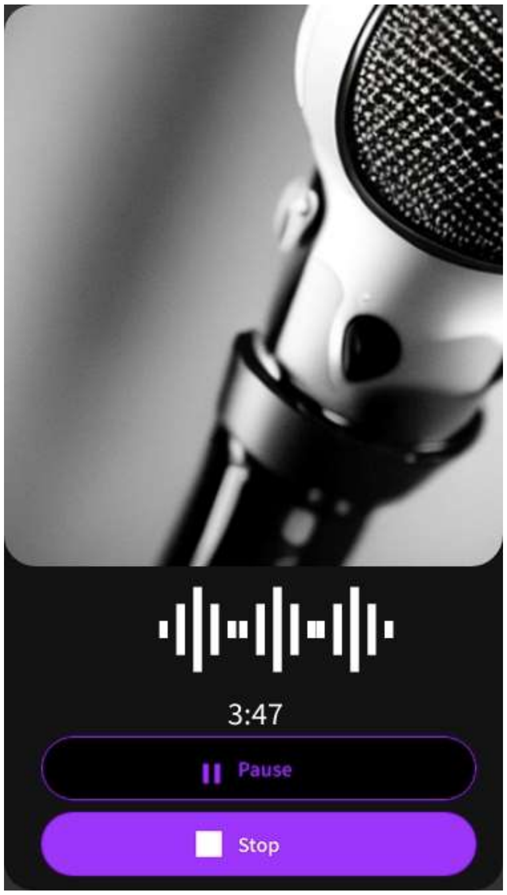


Figure 14: Recording - Started Screen

4.1.9. Recording – Paused Screen

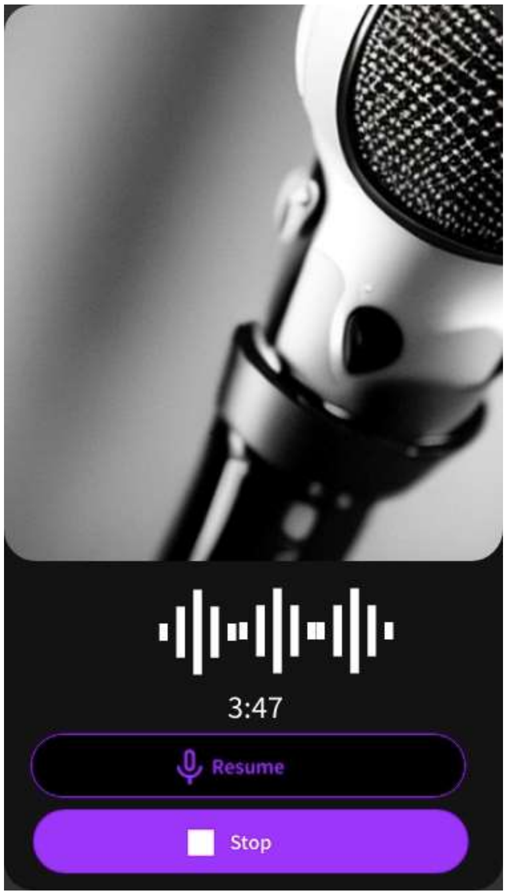


Figure 15: Recording - Paused Screen

4.1.10. Recording – Stopped Screen

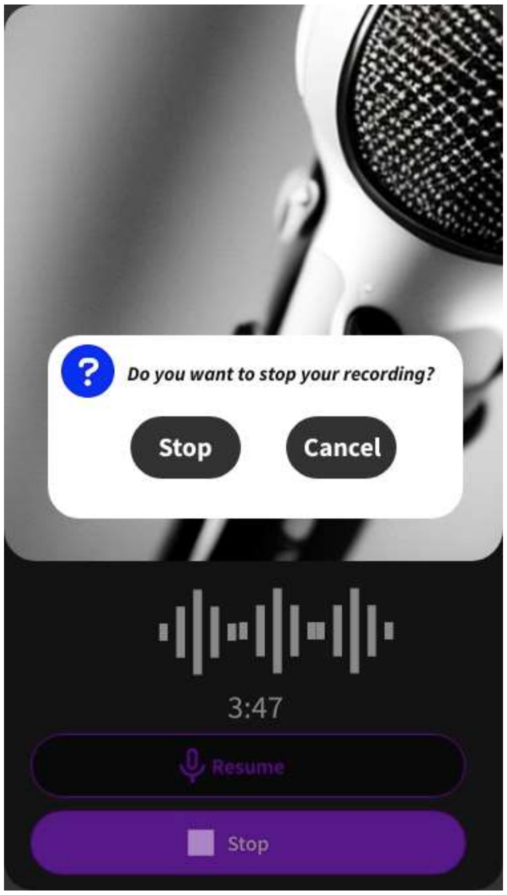
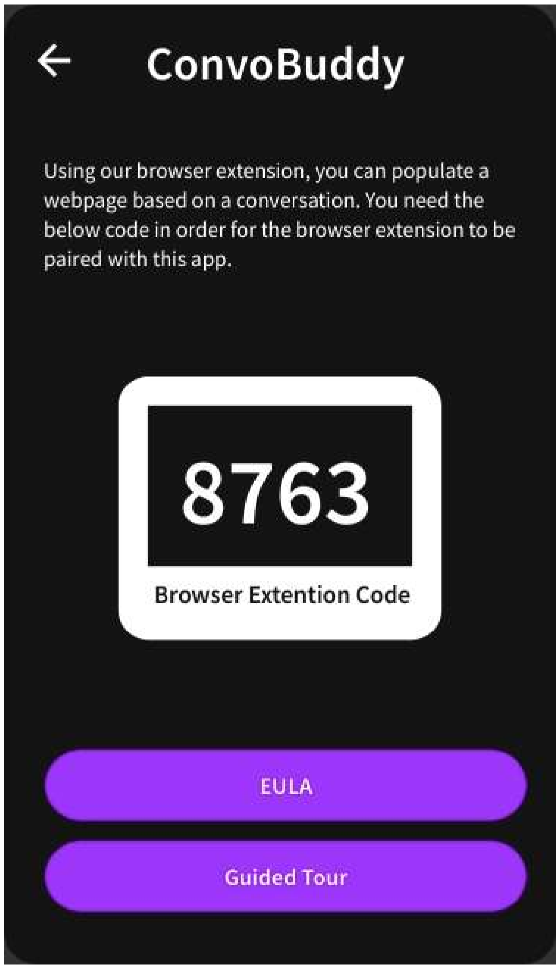


Figure 16: Recording - Stopped Screen

4.1.11. Information Screen


ConvoBuddy

Using our browser extension, you can populate a webpage based on a conversation. You need the below code in order for the browser extension to be paired with this app.

8763

Browser Extention Code

Figure 17: Information Screen

4.1.12. Guided Tour – First Screen

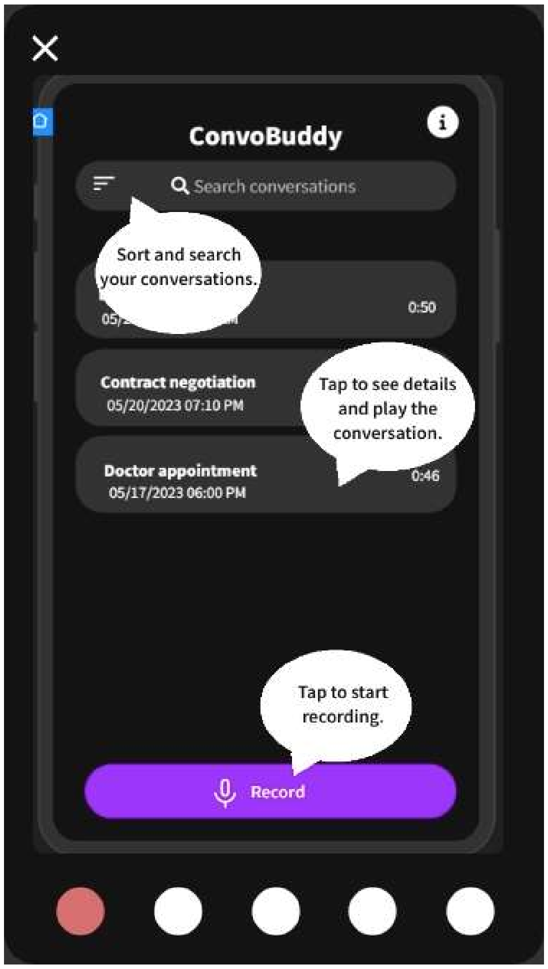


Figure 18: Guided Tour - First Screen

4.1.13. Guided Tour - Second Screen

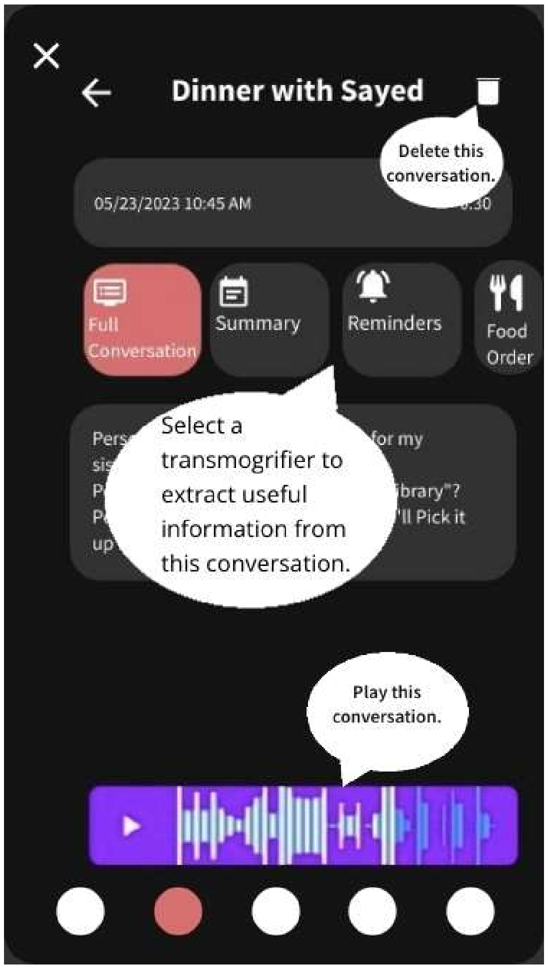

Dinner with Sayed
Delete this conversation.
05/23/2023 10:45 AM
Full Conversation
Summary
Reminders
Food Order
Select a transmogrifier to extract useful information from this conversation.
Play this conversation.

Figure 19: Guided Tour - Second Screen

4.1.14. Guided Tour – Third Screen

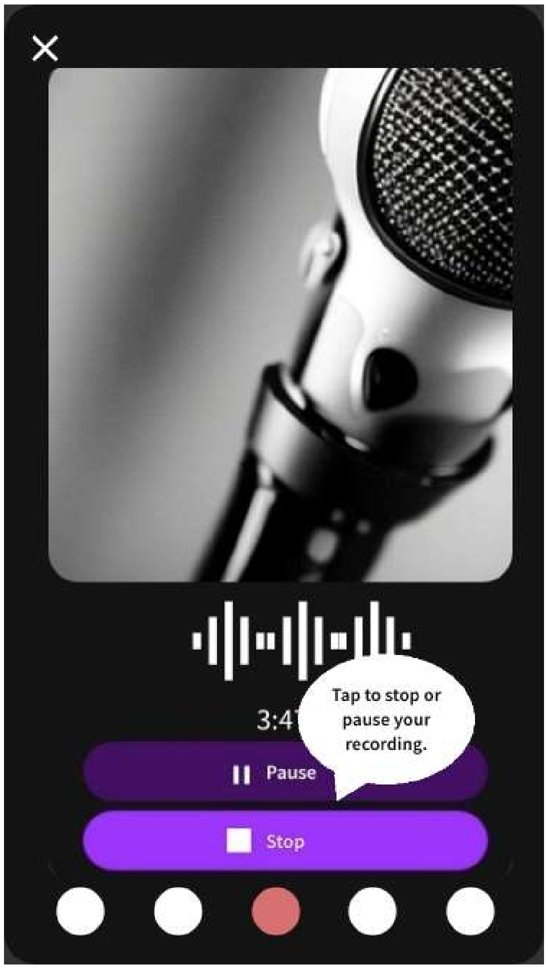


Figure 20: Guided Tour - Third Screen

4.1.15. Guided Tour – Fourth Screen

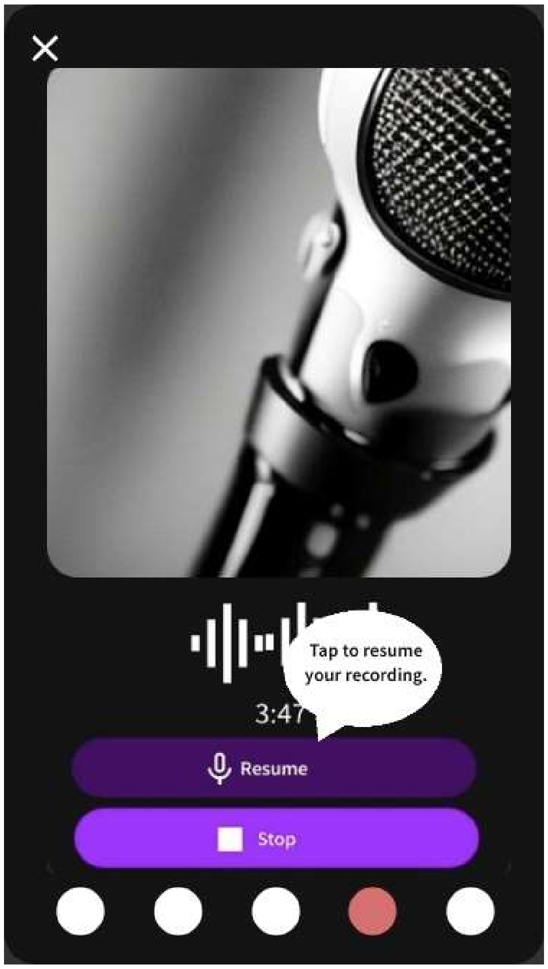
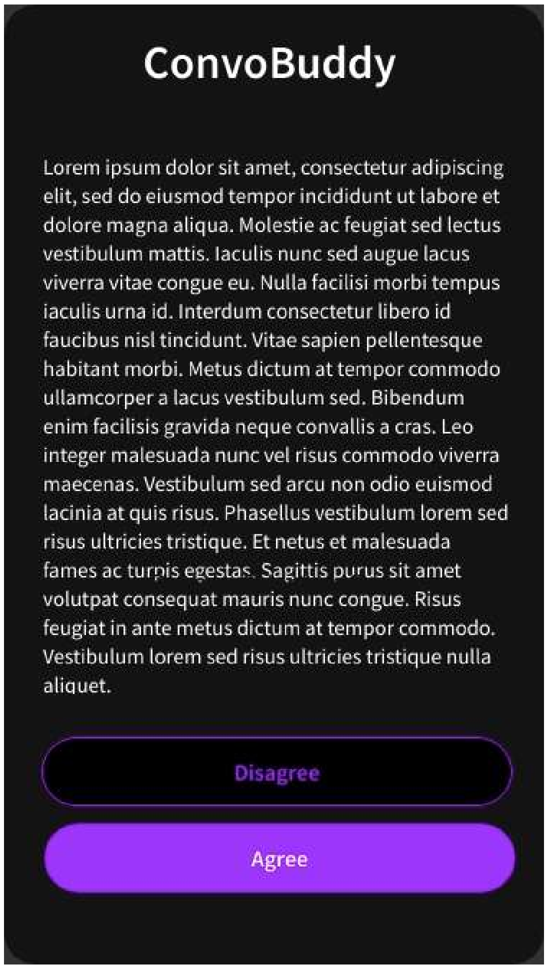


Figure 21: Guided Tour - Fourth Screen

4.1.16. EULA Screen

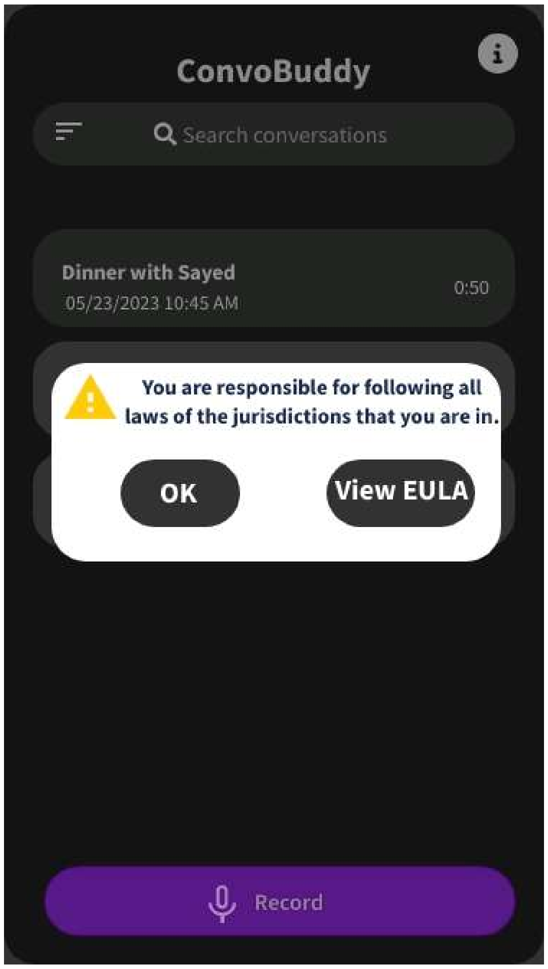


ConvoBuddy

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Molestie ac feugiat sed lectus vestibulum mattis. Iaculis nunc sed augue lacus viverra vitae congue eu. Nulla facilisi morbi tempus iaculis urna id. Interdum consectetur libero id faucibus nisl tincidunt. Vitae sapien pellentesque habitant morbi. Metus dictum at tempor commodo ullamcorper a lacus vestibulum sed. Bibendum enim facilisis gravida neque convallis a cras. Leo integer malesuada nunc vel risus commodo viverra maecenas. Vestibulum sed arcu non odio euismod lacinia at quis risus. Phasellus vestibulum lorem sed risus ultricies tristique. Et netus et malesuada fames ac turpis egestas. Sagittis purus sit amet volutpat consequat mauris nunc congue. Risus feugiat in ante metus dictum at tempor commodo. Vestibulum lorem sed risus ultricies tristique nulla aliquet.

Figure 22: EULA Screen

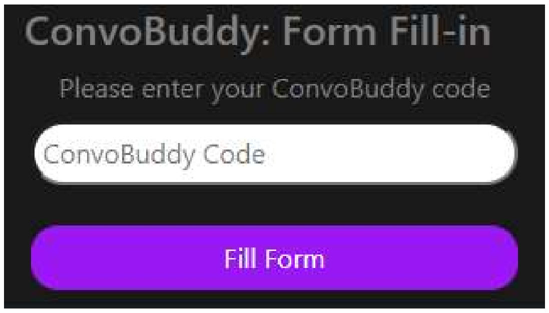
4.1.17. EULA – First Launch Screen



You are responsible for following all laws of the jurisdictions that you are in.

Figure 23: EULA – First Launch Screen

4.1.18. Browser Extension



Please enter your ConvoBuddy code

Figure 24: Browser Extension