Short-Term Memory System (STeMS) Technical Design Document By AlphaSoft

Team Members

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
17 June 2023	1.0	Initial submission to Client	AlphaSoft
22 July 2023	2.0	Updated to match current state of project.	AlphaSoft
5 August 2023	3.0	Updated to match current state of project.	AlphaSoft

Reviewer(s)

Name	Title
Matt Bond	Technical Writer Lead
Mackenzie Cater	Technical Writer
Max McCool	Technical Writer
Taylor McLaughlin	Technical Write

Approver(s)

Name	Title	Approved Date
Matt Bond	Technical Writer Lead	16 June 2023
Oleksiy Blavat	Developer Lead	16 June 2023
Max McCool	Business Analysis Lead	16 June 2023
Michael Powers	Team Lead	16 June 2023

Table of Contents

Figures & Tables

1. Introduction

1.1. Purpose

This Technical Design Document (TDD) describes the architecture and system design of the Short-Term Memory System (STeMS). The full system architecture, including both functional and non-functional requirements, dependencies, workflow, and risk assessments are provided in subsequent sections of this document. This document also provides the user interface and data design elements for both the mobile application and browser extension portions of STeMS.

1.2. Intended Audience

This document is intended to be used by the AlphaSoft Team as a reference for the design and features required by STeMS. The AlphaSoft Quality Assurance Team will use this TDD to assist in the creation of a Test Plan document and subsequent test cases as a means of verifying and validating the work performed by the development team. This document will also be accessible by the Project Manager and Project Owner as reference when ascertaining the current state of the project.

1.3. Project Documents

This TDD will be included in the documentation portion of the project deliverables. All documentation created throughout the project lifecycle is done so with the intent to assist in the understanding, implementation, and maintenance of both the mobile application and browser extension.

The following documents will be included as part of the project deliverables.

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
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 TDD, 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

Term	Definition
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
AWS	Amazon Web Services, a cloud provider of many sundry services.
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
ConvoBuddy	The app under development by this project
DOM	Document Object Model
EULA	End User License Agreement
Flutter	A software framework for developing cross-platform mobile applications
HTTP	Hypertext Transfer Protocol
iOS	iPhone Operating System
JSON	JavaScript Object Notation
Mobile Device	A smart phone, tablet, or some other portable computer with either the iOS or Android operating system
LLM	Large Language Model, a type of AI algorithm
PG	Programmer's Guide
PP	Project Plan
Deployment and Operations Guide	A document that explains how to deploy the program
SRS	Software Requirements Specification
STeMS	Short-Term Memory System, the name of this project
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
TDD	Technical Design Document
TM	Traceability Matrix. A document that traces defects and test cases back to their requirement.

Term	**Definition**
TMGR	Transmogrifier
TR	Test Report
UG	User Guide
UI/UX	User Interface / User Experience
URL	Uniform Resource Locator, typically a path to files on a computer

Table 2: Acronyms, Definitions and Abbreviations

2. Overview

2.1. Business Requirements Summary

2.1.1. Security Constraints

- 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.
- 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.

2.1.2. Interface constraints

- The application will run on iOS and Android mobile devices.
- The app will only run in vertical mode (portrait) with the following resolution baselines: 1170 x 2532 for iPhone 13 Pro and 1080 x 2280 for Google Pixel 4.
- 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.

2.1.3. 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 search feature allowing searching all of the conversations' text.

2.2. In Scope

The scope of this TDD includes the design plans for the user interface and the user interaction aspects of STeMS down to the component and workflow level. The application will allow users to create and manage live audio recordings and will employ ChatGPT to analyze those recordings and provide additional functionality. This technical document also outlines an additional browser extension that will allow the application to use processed recordings to populate webpage forms. This TDD covers only the front-end features for the application.

Additionally, this document will outline the UI styles decided by the team that align with the goals of the application and the users' needs. It may also cover the minimal UI required for the browser extension to function in those other programs.

2.3. Out of Scope

This document will not address any back-end function, design, or support required by the UI, unless it is displayed to or modifiable by the user. Functional interaction with the browsers or other programs will likewise not be included. All back-end implementation is outside the responsibility of the AlphaSoft Development team and will be outlined in a separate document by Team B.

3. System Overview

3.1.Summary

The main purpose of STeMS is to help users recall important parts of their conversations by giving them the tools to record, review, and use ChatGPT to act upon them. To this end, the UI of ConvoBuddy shall be easy to learn, easy to see, and easy to use in daily life by all users, with a special focus on those with memory or hearing impairments. The design shall minimize delays and other obstacles where possible that users may face when using the app.

3.2.Technical Approach

Development shall proceed from the current draft of the UI, which includes mockups of the app's screens. Because AlphaSoft is responsible for the front end, the application will be graphically designed, and then functionality will be added to each element subsequently before being tied to the back end. There are some elements that have been tested out already, including parts of the Browser Extension, but now that they are confirmed to be working the graphics will be prioritized. The two main screens are the Conversation List and the Conversation Details screens. It is from these two that most other actions and views originate, so development will begin with those to allow the most time for improvement and not block downstream workflow processes. After these two screens, the recording and transmogrifier screens should be next, followed by the Information and Guided Tour screens, and finally the message/error pop-ups.

3.3.Workflow

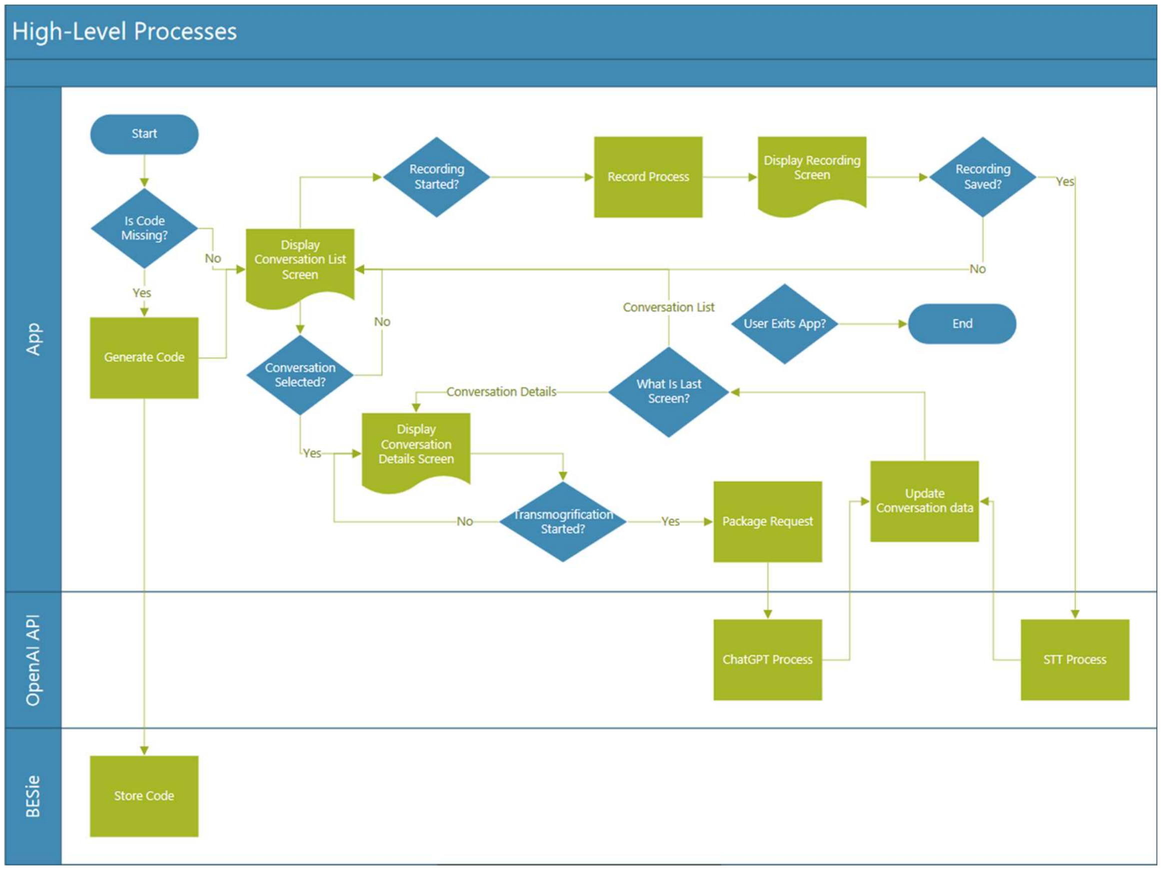


Figure 3: High-level workflow of STeMS

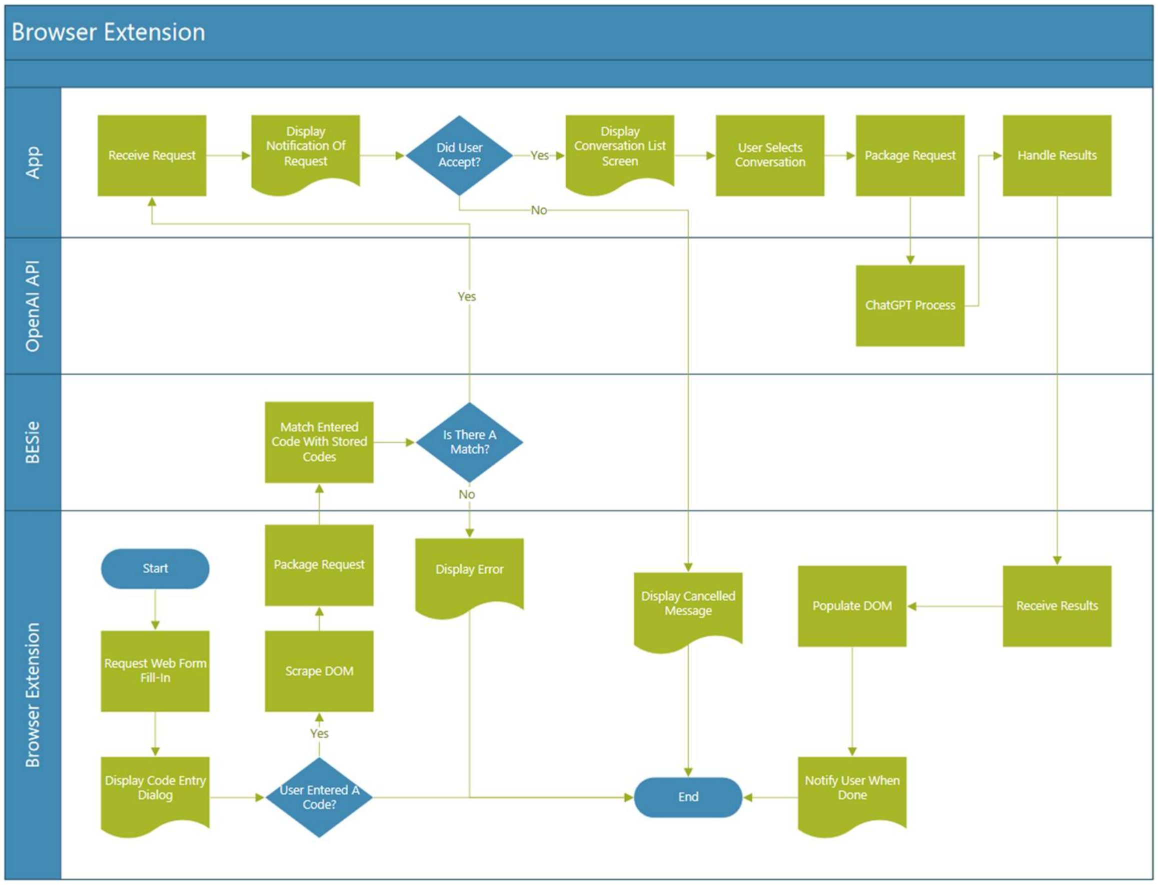


Figure 4: Workflow of the browser extension and related functionality

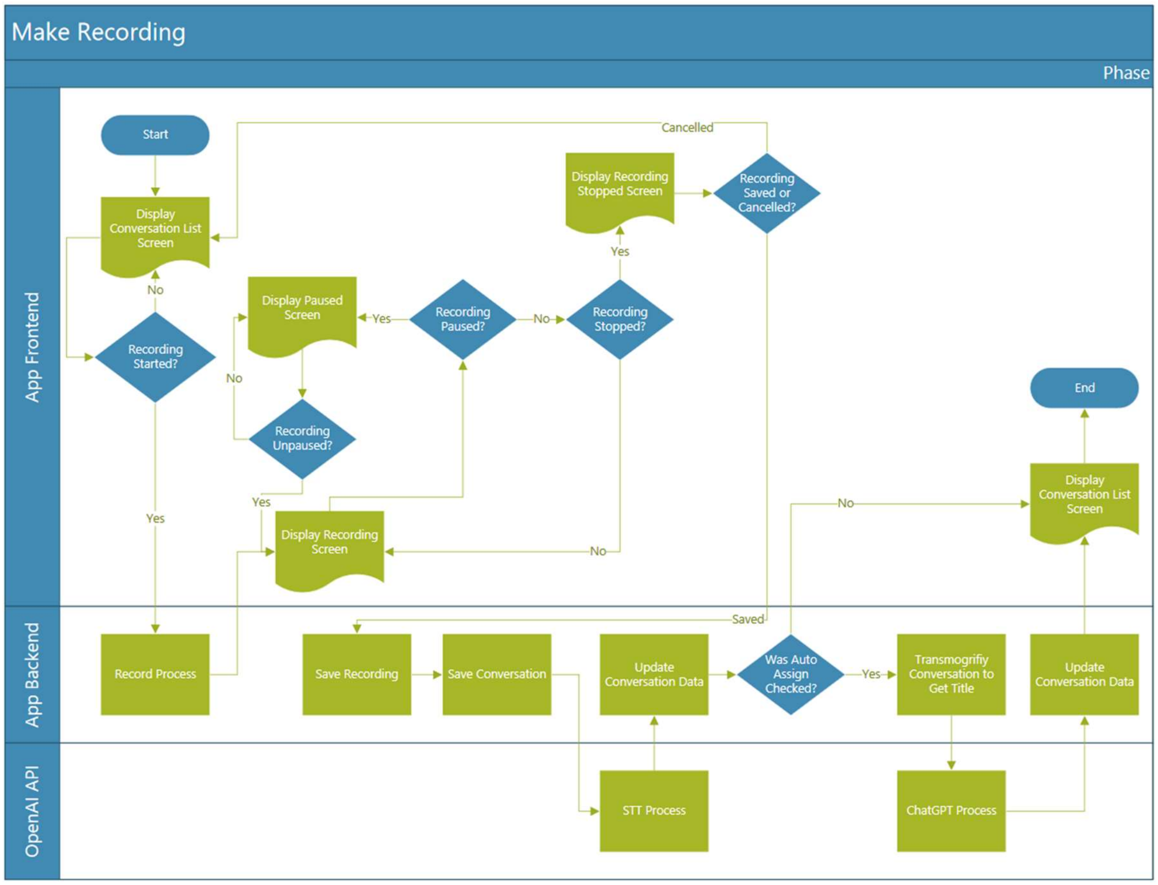


Figure 5: Workflow of recording a conversation

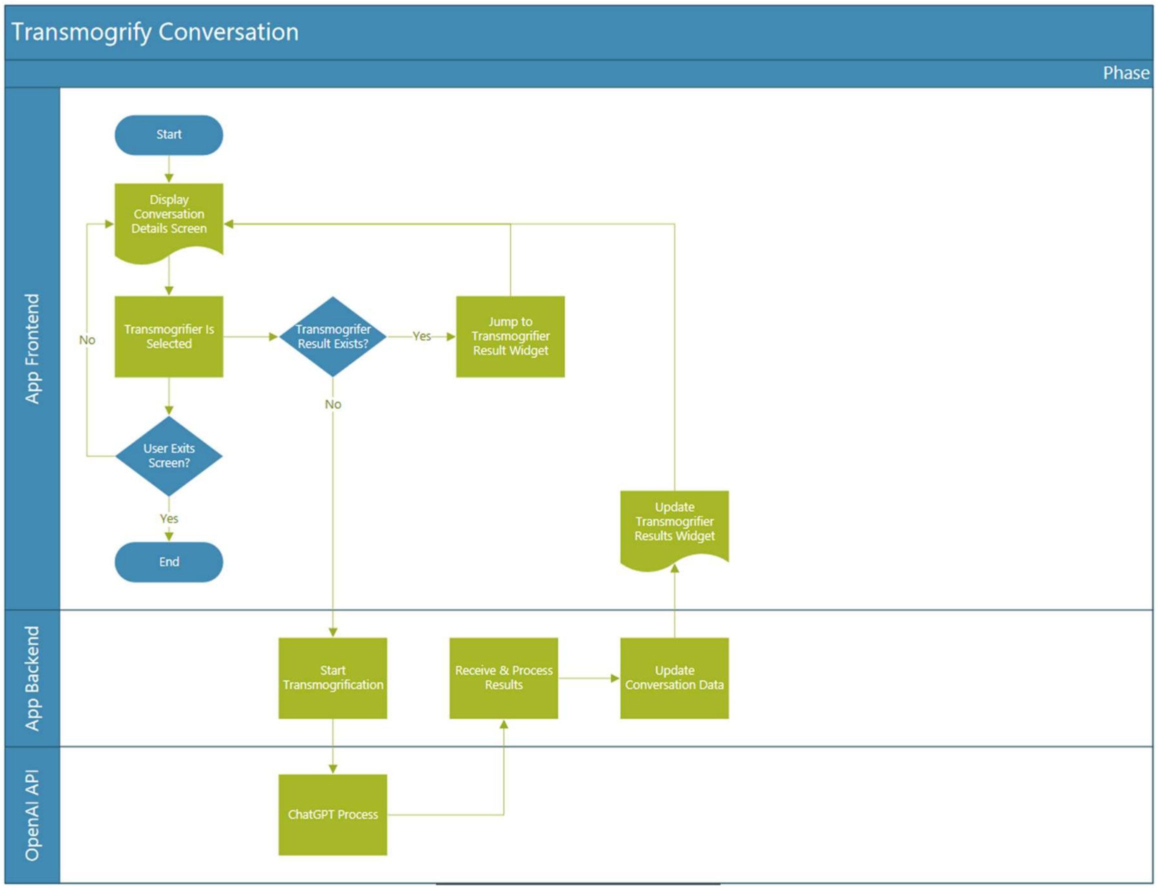

Transmogrify Conversation
Phase
App Frontend
App Backend
OpenAI API
Start
Display Conversation Details Screen
Transmogrifier Is Selected
Transmogrifer Result Exists?
Yes
No
Jump to Transmogrifier Result Widget
No
User Exits Screen?
Yes
End
Update Transmogrifier Results Widget
Start Transmogrification
Receive & Process Results
Update Conversation Data
ChatGPT Process

Figure 6: Workflow of the transmogrifying process

3.4. Architecture

This section clarifies the system architecture of STeMS, an application that is composed of three key components, tasked with accomplishing four key use cases. The system's development involves the collaboration of two distinctive teams: AlphaSoft, which is responsible for the front-end layer, while Team B oversees the back-end of the application, ensuring the efficient management and operation of data. AlphaSoft's primary focus is on crafting a user-friendly, intuitive experience, which is essential for serving the main user demographic – individuals grappling with short-term memory loss.

The mobile app is built upon Flutter, a cross-platform mobile framework, guaranteeing compatibility with both iOS and Android platforms. The application offers a suite of robust features, including the capability to record and transcribe conversations, fill out online forms, and assist individuals with short-term memory challenges by extracting critical, time-sensitive information from their recorded conversations.

The functionality of STeMS is powered by five primary components: the cross-platform mobile application (ConvoBuddy), a browser-agnostic extension, an intermediary service known as the Back End Service (BESie), integration with AWS's Amazon Transcribe STT service with diarization, and integration with OpenAI's ChatGPT API. BESie plays a central role in communicating with ConvoBuddy, Amazon Transcribe, and the browser extension. The convergence of these components positions STeMS as a highly efficient tool, enhancing daily productivity and communication capabilities for its users.

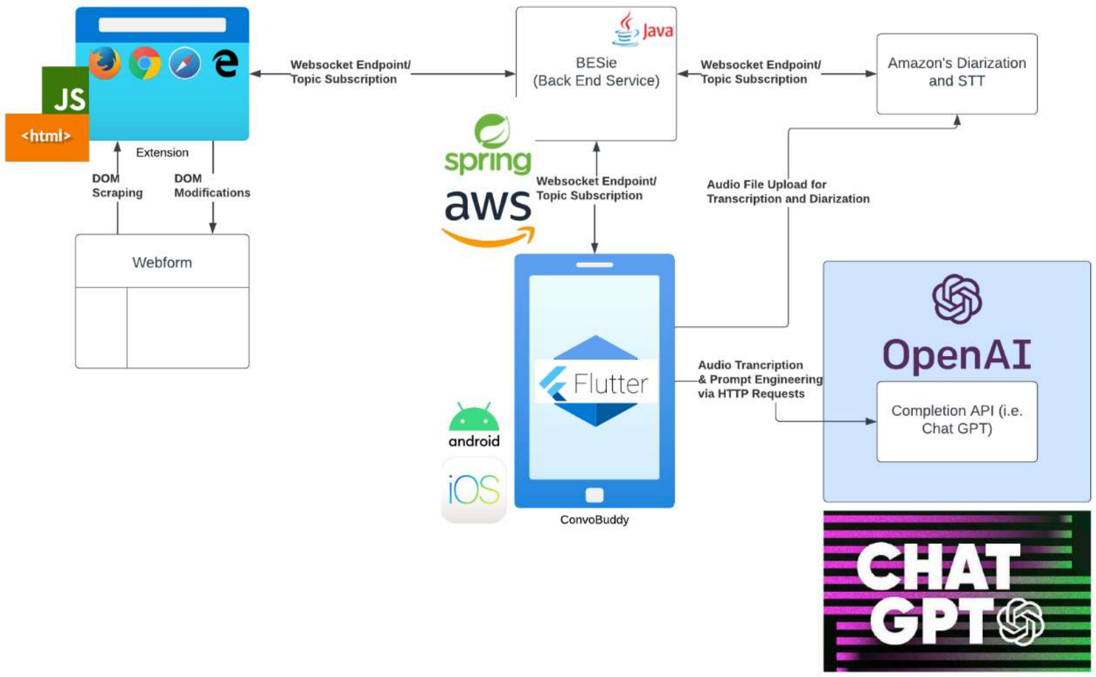

Extension
Websocket Endpoint/ Topic Subscription
BESie (Back End Service)
Java
Websocket Endpoint/ Topic Subscription
Amazon's Diarization and STT
DOM Scraping
DOM Modifications
Webform
spring
aws
Websocket Endpoint/ Topic Subscription
Audio File Upload for Transcription and Diarization
Flutter
Audio Trancription & Prompt Engineering via HTTP Requests
OpenAI
Completion API (i.e. Chat GPT)
android
iOS
ConvoBuddy
CHAT GPT

Figure 7: ConvoBuddy Components and high-level interaction

3.5. Decomposition

3.5.1. Form Filler Decomposition

ConvoBuddy boasts a feature that allows for intuitive online form completion, fostering a smooth user experience. This feature is activated when a user navigates to an online form in their browser and chooses the “Fill Form” option via ConvoBuddy's browser extension. The user then switches to the ConvoBuddy app to select a suitable conversation that contains the relevant information for the form.

The seamless interaction between the ConvoBuddy browser extension and the app is achieved through a WebSocket connection with defined endpoints and topics. The browser extension scrapes the Document Object Model (DOM) of the webpage for fillable fields, composing a JSON payload which is consumed by a WebSocket service. This service is a part of BESie.

BESie is equipped with two endpoints and two topics, facilitating effective communication between the browser extension and the mobile app. Once the extension has prepared the form description payload, it sends it to the “fill” endpoint of BESie. This service, in turn, broadcasts the message to a topic named 'fill form', which the app is continually monitoring.

Upon detecting a new message on the “fill_form” topic, ConvoBuddy parses the payload and generates a push notification for the user. The user interacts with the notification, selecting a conversation, and the corresponding transcription along with the form description is sent to OpenAI's completion endpoint. The challenge lies in crafting a precise prompt that enables OpenAI to scan the conversation transcript and return a JSON response with the keys representing fillable fields and the values containing corresponding information for those fields.

This result is then sent to the browser extension via the WebSocket connection, with the extension awaiting this data at the “filled_form” endpoint. Upon receiving the payload from BESie, the extension populates the appropriate fields in the DOM, simplifying and expediting the form-filling process for the user.

Example payload for a contact form:

```
{{
  "form": [
    {
      "include_email": {
        "inputType": "checkbox",
        "inputValue": "on"
      }
    },
    {
      "message": "text"
    },
    {
      "feedback": "textarea"
    },
    {
      "user[profile_pronouns]": {
        "inputType": "select",
        "optionList": [
          "",
          "they/them",
          "she/her",
          "he/him",
          "Custom"
        ]
      }
    },
  ],
  "pin": "4876"
}
```

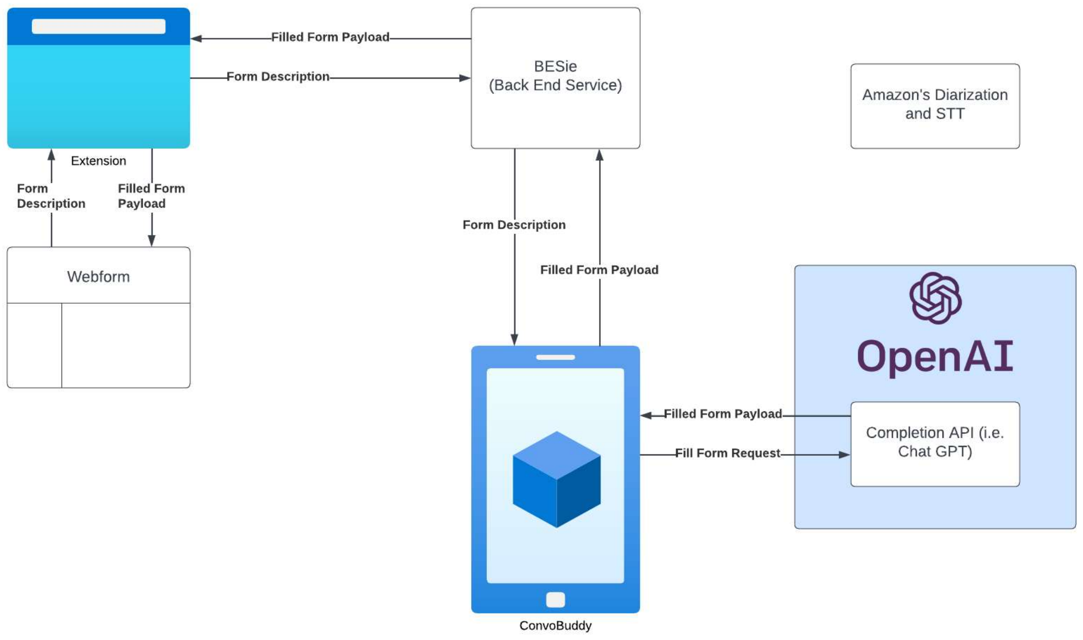


Figure 8: Form Filler Decomposition

3.5.2. Transmogrifier Decomposition (Generic)

The term 'transmogrifier' has been creatively adapted by AlphaSoft to denote the array of features that ConvoBuddy offers, powered by artificial intelligence (AI). These transmogrifiers (TMGR) facilitate a transformation, altering the data input to generate meaningful and useful output.

A core transmogrifier is the Reminder transmogrifier. This feature is activated at the end of each conversation recording. Upon termination of the recording, the raw audio file is stored on the user's device and is dispatched to AWS's Amazon Transcribe, a STT service with diarization (implemented by Team B). This tool transcribes the audio into human-readable text, considering aspects such as removal of filler words for enhanced readability. It can also distinguish between different speakers.

Upon completion of the transcription, a second request is directed to Open AI's completion endpoint. This request is accompanied by a carefully crafted prompt designed to extraction important information from the conversation, such as follow-up items, appointments, or promises made. This summary information is displayed in the conversation detail screen, saving users the effort of sifting through lengthy transcripts.

The potential for expansion of these transmogrifiers is vast. Currently, the primary transmogrifiers are "take order", which assists waitstaff in noting down table orders directly from the customers' conversations; "reminder", which auto-activates after every recorded conversation; and "form filler", as previously detailed. These transmogrifiers enable ConvoBuddy to evolve from a mere recording tool into a comprehensive assistant, interpreting and responding to a user's needs effectively.

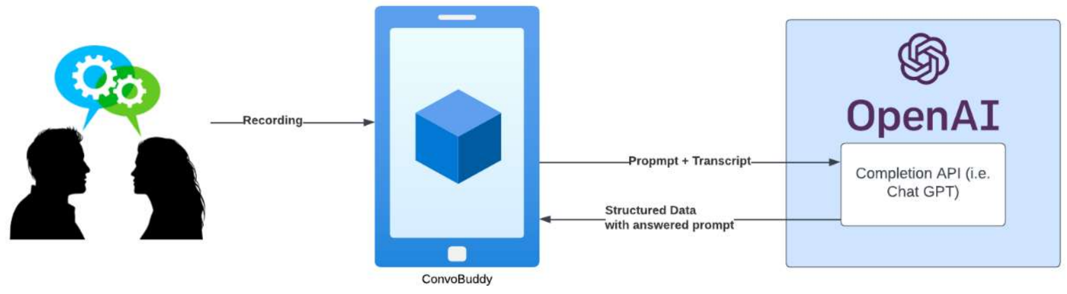

Recording
ConvoBuddy
Propmpt + Transcript
Structured Data with answered prompt
OpenAI
Completion API (i.e. Chat GPT)

Figure 9: Transmogrifier Generic Decomposition

3.6. Functional Requirements

The following table shows the breakdown of functional requirements for STeMS. The functional requirements are meant to identify what ConvoBuddy and the supporting components of the overall system should do to satisfy the business use cases of the application.

Id	Requirement	Description
1	Recording Conversation	The app should be able to record conversations with clear audio quality.
2	Conversation Transcription	After the conversation is recorded, the app should be able to transcribe the conversation using Amazon Transcribe.
3	Form Filling	The app should have the ability to auto-fill online forms. It should interact with a browser extension to identify fillable fields and populate them based on past conversations.
4	Reminder Generation	The app should automatically generate reminders, follow-up items, and appointments from the transcribed conversation text.
5	Intermediary Service Communication	The app must maintain a WebSocket connection with the intermediary service (BESie), which facilitates communication between the browser extension and the app itself.
6	Notification	The app should send push notifications to the user for any new form fill requests.
7	Integration with OpenAI Products	The app should seamlessly integrate with Amazon Transcribe for transcription and ChatGPT for processing and understanding transcriptions.
8	Storing Conversation Data	The app should store the raw audio file on the user's device and also keep a record of the transcribed text for future reference.
9	User-Friendly Interface	The app should offer an intuitive and simple user interface, particularly for individuals with short-term memory loss.
10	Cross-Platform Compatibility	The app should be compatible with both iOS and Android platforms.

Id	Requirement	Description
11	Order Taking	The app should support a 'take order' function to help waiters understand and record table orders from customer conversations.
12	Data Security	The app should securely handle and store user data, ensuring privacy and confidentiality. All voice recordings are to be stored on the user's device only.

3.7.Non-Functional Requirements

The table below outlines the non-functional requirements for STeMS. Non-functional requirements will describe how STeMS should behave and set certain standards for the system's overall performance.

Id	Requirement	Description
1	Performance	The app must transcribe audio to text promptly and fill out forms accurately and swiftly to provide real-time assistance to users. The WebSocket connection should maintain low latency for quick and efficient communication.
2	Reliability	The app must consistently perform its functions under stated conditions without failure. It should also have a high uptime and ensure recorded conversations are not lost.
3	Usability	The user interface of the app must be intuitive and straightforward to cater to users with short-term memory loss. It should also provide a comfortable user experience for all kinds of users, including waiters and regular individuals.
4	Security and Privacy	All user data, including audio recordings, transcriptions, and personal information used to fill forms, should be securely stored and handled in accordance with relevant data protection laws and regulations. The WebSocket connection should be secured to prevent any data breaches.
5	Scalability	The app should be capable of handling a larger number of users or an increased volume of data without impacting performance or functionality.
6	Compatibility	The app should be compatible with the latest versions of iOS and Android operating systems, and the browser extension should work seamlessly across different browsers.
7	Maintainability	The app should be easy to modify and update to improve its performance, correct defects, or adapt to changes in the environment or requirements.
8	Resilience	The app should have the ability to recover quickly from any failures, ensuring minimal disruption to the user.
9	Interoperability	The app should easily integrate and work with third-party systems, including AWS's Amazon Transcribe and OpenAI's ChatGPT.

Id	Requirement	Description
10	Availability	The app should be available for use as consistently as possible, with minimal downtime.

3.8.External Dependencies

3.8.1. Third-Party Components

The mobile application depends on the following third-party Flutter packages to speed up development:

- Audio_waveforms – Provides a customizable soundwave widget paired with an audio recording interface
- Path_provider – Provides generic functions to find app-specific folders without needing to write specific iOS and Android platform code.
- Stop_watch_timer – Provides stopwatch functionality that extends the basic Flutter stopwatch by emitting events at specified intervals.

3.8.2. Vendors

STeMS, as a system, depends on the following vendors for its functionality

- OpenAI – Open AI provides a robust API with numerous endpoints. To satisfy business functionality, ConvoBuddy relies on Open AI's transcription and completion endpoints. The transcription endpoint is responsible for converting audio files into text. The completion endpoint is responsible for answering complex prompts which leverage Open AI's large language model (LLM).

3.9.Assumptions

- API endpoints will not change their request and response object structures for the initial release.
- Users will install and run the mobile application on a device that is running at least Android version 11 or iOS version 12.
- Users will have their device connected to the internet via Wi-Fi, mobile network, etc.
- Most users will not be using the app between 12 AM and 1 AM EST and thus updates to any parts of the system will be pushed at that time to cause the least amount of disruption.
- The user base will be small for the initial release so scalability is not a concern (i.e., moving to cloud storage, multiple servers, etc.).
- Everyone on the team will be available to test the product before release.
- Everyone on the team has the necessary tools installed to run and test the software locally.

3.10. Issues

- Developer inexperience with the Flutter framework and Dart programming language.
- Developer inexperience creating browser extensions.
- Small time frame to implement and adequately test all feature requirements.
- The team does not have an equal number of iOS and Android devices to adequately test the mobile application on both operating systems.
- The cost for having everyone on the team develop against the API is not beneficial, and thus API related work can only be completed and tested by a few team members.

3.11. Risks

- Android and iOS system updates could break or negatively affect the core functionality of the mobile app. The team will stay up-to-date with each Android and iOS release to ensure code changes are not needed.
- Android and iOS system updates could break third-party Flutter packages that the application depends on. The team will stay up-to-date with each Android and iOS release and check the third-party packages' GitHub pages to ensure breaking changes are not introduced.
- API endpoints could change their URLs, accessibility, request/response packages, etc., which could render the mobile app's core functionality useless. To mitigate this, the team needs to check with the API's documentation monthly to ensure there are no upcoming changes.
- Scope creep could compromise the deadline for product delivery. The team will verify any new requirements are not already met by previous requirements, and then create and present a plan to the customer on why the new requirement can or cannot be implemented in the given timeframe.

4. Design Elements for Mobile Application

4.1.Data Design

4.1.1. Summary

The following list of objects are used within the mobile application to store data that will be displayed on the UI, sent to an API, or retrieved from API. The objects do not persist in memory once the application is closed.

4.1.2. Objects

Object Name	Object Type	Purpose
SortingType	Enum	Contains the types of sorting that can be applied onto a list of conversations.
TransmogType	Enum	Contains the types of transmogrifiers that can be applied to a conversation.
Conversation	Class	Holds a conversation's metadata as well as the results of any transmogrifier processes.
TakeOrderRequest	Class	Contains conversation data needed for the API to process the "Take Order" transmogrifier.
TakeOrderResponse	Class	Contains result data returned from the API after the "Take Order" transmogrifier is completed.
FormFillerRequest	Class	Contains conversation data needed for the API to process the "Form Filler" transmogrifier.
FormFillerResponse	Class	Contains result data returned from the API after the Form Filler" transmogrifier is completed.
TranscriptionRequest	Class	Contains conversation data needed for the API to complete the speech-to-text transcription process.
TranscriptionResponse	Class	Contains result data returned from the API after the transcription process is completed.

4.1.2.1. Object: SortingType Enum

Values

- dateNewToOld,
- dateOldToNew,
- titleAToZ,
- titleZToA,
- durationShortToLong,
- durationLongToShort

4.1.2.2. Object: TransmogType Enum

Values

- formFiller
- reminder
- takeOrder
- transcription

4.1.2.3. Object: Conversation

Properties

Name	Data Type	Required on Creation	Summary	Purpose
id	String	Yes	A randomly generated GUID string at time of creation.	To uniquely identify a conversation.
title	String	Yes	"Conversation_" and an incrementing number. Ex: "Conversation_2".	To populate UI title element of a conversation
recordedDate	DateTime	Yes	The date that the audio recording of a conversation was created.	Used for sorting conversations by date.
duration	Duration	Yes	The duration of the audio recording.	Used for sorting conversations by duration.
audioFilePath	String	Yes	The location of the audio recording on the device.	Used to quickly find the audio file to upload for transcription and playback purposes.
content	String	No	Will contain a conversation's transcription after the Transcription transmogrifier is completed.	To store a recording's transcript if and when a transcription process was executed on a conversation.
transmogs	Map<TransmogType, String>	No	The map will be initially filled with each transmogrifier type representing a key and the value is an empty string.	To store each result for a specific transmogrifier.

4.1.2.4. Object: TakeOrderRequest

Properties

Name	Data Type	Required on Creation	Summary	Purpose
conversationId	String	Yes	The ID of the conversation that this transmogrifier is associated with.	To keep track of which conversation this request is associated with.
conversationTranscript	String	Yes	A conversation's content property value.	Required for the API's purpose of parsing a transcript with the "Take Order" transmogrifier.

4.1.2.5. Object: TakeOrderResponse

Properties

Name	Data Type	Required on Creation	Summary	Purpose
conversationId	String	Yes	The ID of the conversation that this transmogrifier is associated with.	To find and update the conversation this transmogrifier is associated with.
result	String	Yes	Contains the result data the API returned for the "Take Order" transmogrifier.	To show the user the results of executing the "Take Order" transmogrifier on a conversation.

4.1.2.6. Object: FormFillerRequest

Properties

Name	Data Type	Required on Creation	Summary	Purpose
conversationTranscript	String	Yes	A conversation's content property value.	Required for the API's purpose of parsing a transcript with the form filler transmogrifier.
formDescription	Map<String, String>	Yes	Contains fillable properties found on a form via the ConvoBuddy browser extension.	Required for the API to determine what data to look for in a conversation transcript.

4.1.2.7. Object: FormFillerResponse

Properties

Name	**Data Type**	**Required on Creation**	**Summary**	**Purpose**
formDescription	Map<String, String>	Yes	Contains fillable properties found on a form via the ConvoBuddy browser extension.	Needed to send back to the API in a FormFillerRequest.

4.1.2.8. Object: TranscriptionRequest

Properties

Name	**Data Type**	**Required on Creation**	**Summary**	**Purpose**
conversationId	String	Yes	The ID of the conversation that this transcription is associated with.	To keep track of which conversation this request is associated with.
audioFilePath	String	Yes	A conversation’s audioFilePath property value.	To let the user’s device know the path of the file to upload for transcription.

4.1.2.9. Object: TranscriptionResponse

Properties

Name	**Data Type**	**Required on Creation**	**Summary**	**Purpose**
conversationId	String	Yes	The ID of the conversation that this transcription is associated with.	To find and update the conversation this transcription is associated with.
content	String	Yes	The result from the Transcription endpoint that is a text version of a conversation’s recorded audio.	To populate a conversation’s content property.

4.2.UX Design

4.2.1. Components

All Flutter components are of the Widget type. Some widgets are custom built for the project while others are provided by the Flutter framework and third-party packages.

When a component is used on multiple screens and is not a base Flutter widget, it will be extracted into a reusable custom widget.

Component	**Screen**	**Summary**
Alert Dialog	All screens	A base Flutter widget that displays a pop-up that overlays the current screen and contains important information for the user. It can optionally contain a cancel and confirmation button.
Animated Audio Wave	Recording - Started, Recording - Paused	Customizable widget that generates real-time sound waves as audio is recorded or played.
App Bar	Conversation List, Conversation Details, Information	A toolbar-like widget displayed at the top of a screen that typically provides text and buttons.
Checkbox List Tile	Recording - Stopped	Simple checkbox widget with a label that lets the user choose to auto populate a recording title.
Code Viewer Card	Information	Custom widget containing a code to connect to the browser extension.
Conversation List Item	Conversation List	Custom widget used in a list. Contains detailed information about a conversation and can be assigned an “onPress” action.
Icon Button	Conversation List, Conversation Details	Base Flutter widget for using icons as buttons.
Image	Recording - Started, Recording - Paused, Recording - Stopped	Base Flutter widget that simply displays an image.
Image Carousel	Guided Tour	Custom widget that allows users to swipe through a collection of images.
Search Bar	Conversation List	Custom widget that reacts to user text input and filters a list of conversations based on that input.
Sorter Popup Menu	Conversation List	Custom widget that displays an icon button to be pressed. If and when the button is pressed, a pop-up menu with sorting options is displayed. If and when the user presses and option or outside of the menu, the pop-up menu will close.
Text	Conversation List, Conversation Details, Information, Recording - Started, Recording - Paused, Recording - Stopped	Base Flutter widget to display strings to the user.
Text Button	Conversation List, Conversation Details, Recording Started, Recording Paused, Recording Stopped, Information	Base Flutter widget for displaying text and optional leading icons on a button.
Text Field	Recording - Stopped	Base Flutter widget that allows users to enter text input.
Timer	Recording - Started, Recording - Paused	Custom component that acts as a stopwatch and emits events with its current time at a specified interval.

Component	**Screen**	**Summary**
Transmog0 Category Picker	Conversation Details	Custom widget composed of a list of "Transmog List Item" widgets that can be horizontally scrolled.
Transmog List Item	Conversation Details	Custom widget that represents a single, selectable transmogrifier that can be performed on a conversation.

4.2.2. Message Catalog

All messages to the user within the mobile application will be displayed within the Alert Dialog widget. This widget will overlay onto the current screen and display important text to the user such as the cause for an error or confirmation of an action. If the pop-up requires user input, "Cancel" and "Ok" buttons will be displayed on the pop-up.

Type	**Cause**	**Message**
Warning	No Internet Connection	Unable to connect. Your device may be offline.
Error	Flutter and third-party package exceptions	There was a problem with the application.
Error	APIs return an HTTP error code	The API failed to process.
Error	Could not load JSON file	Could not retrieve conversations from device.
Error	Attempting to play conversation whose audio file was deleted off the device	Could not locate the conversation's recording.
Confirmation	Deleting a Conversation	Are you sure you want to remove this conversation permanently?
Confirmation	Canceling a recording	Are you sure you want to cancel this recording?
Error	Insufficient Device Storage when saving a recording	Cannot save the recording. Insufficient storage.
Error	No Permissions to record audio	Please enable audio recording permissions on your device for this application.
Error	No Permissions for file IO	Please enable read/write permissions on your device for this application.
Error	Insufficient Device Storage when creating a conversation	Cannot save new conversations to device. Insufficient storage.

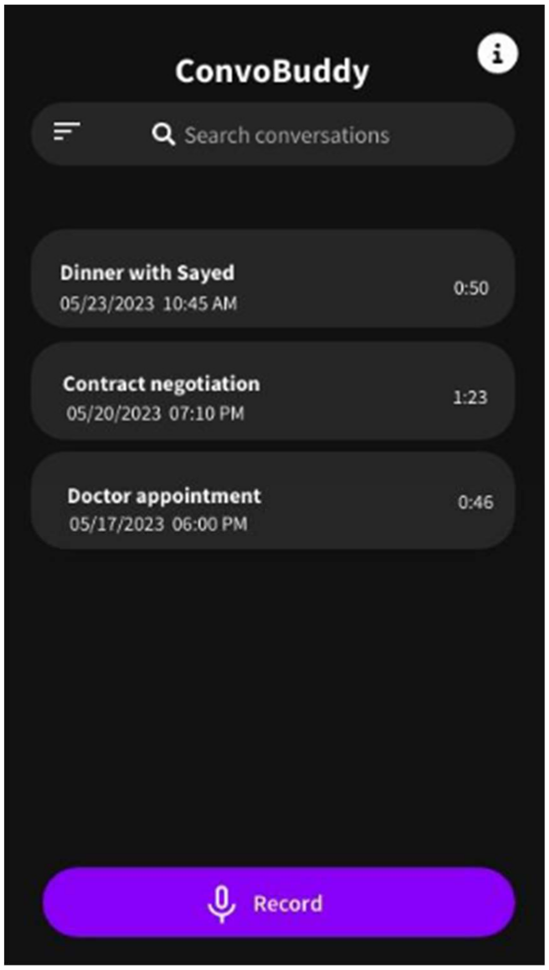
4.3.Screen: Conversation List

4.3.1. Description

This is the starting point of the application. This is the first screen that the users will see once they open the application. On this screen the user can:

- Record a new conversation
- Sort existing conversations
- Search for an existing conversation
- Display the Information screen
- Select a conversation

4.3.2. Images


ConvoBuddy

Search conversations

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

Record

Figure 10: Conversation List Screen

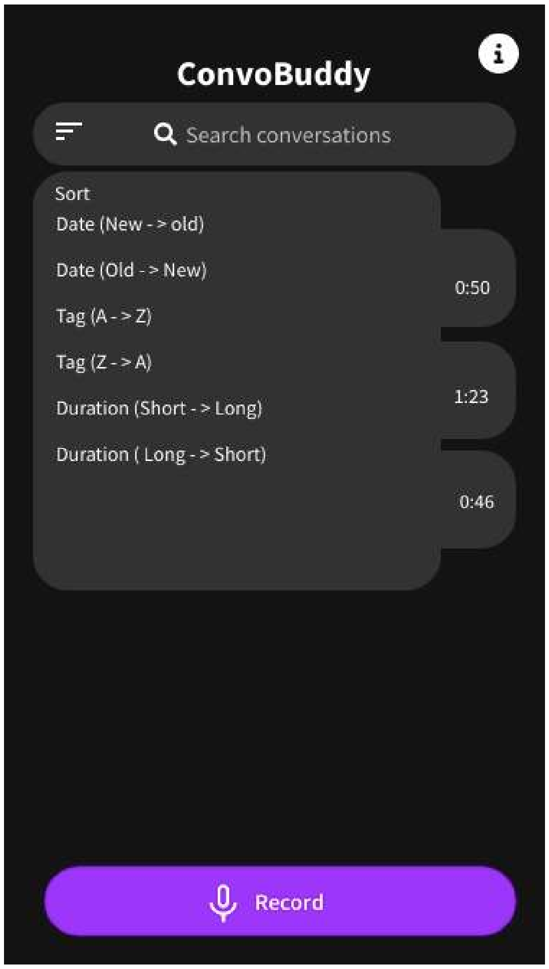


Figure 11: Conversation List - Sort Screen

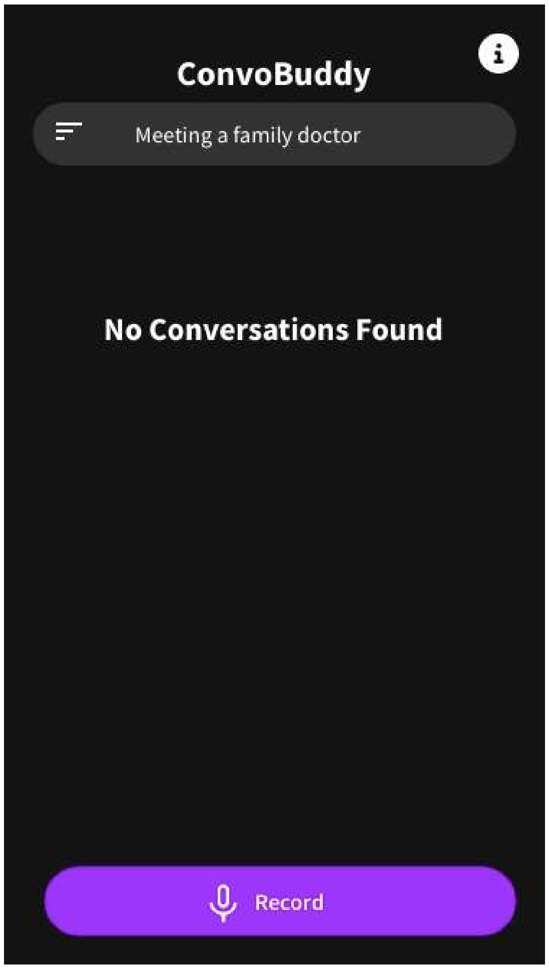


Figure 12: Conversation List - Search Not Found Screen

4.3.3. Components

- App Bar
- Conversation List Item
- Icon Button
- Text Button
- Search Bar
- Sorter Popup Menu
- Text

4.3.4. Internal UI Functionality

- On display, the list of conversations will be loaded, sorted by the last saved sorting option.
- Tapping on the sort icon shall display a sorting options pop-up.
  - Tapping a sorting option shall close the pop-up and then refresh the list of conversations, sorted by the selected option. The sorting option is saved.
  - Tapping outside the sorting option pop-up shall close the pop-up only.
- Tapping into the Search bar shall make the on-screen keyboard appear.

- If the user searches for a conversation, the current list of conversations shall be filtered on the screen or a “No Conversations Found” message shall appear.

4.3.5. External UI Functionality

- Tapping the Record button displays the Recording – Start screen.
- Tapping the Information icon displays the Information screen.
- Tapping on a conversation will display the Conversations Details screen.

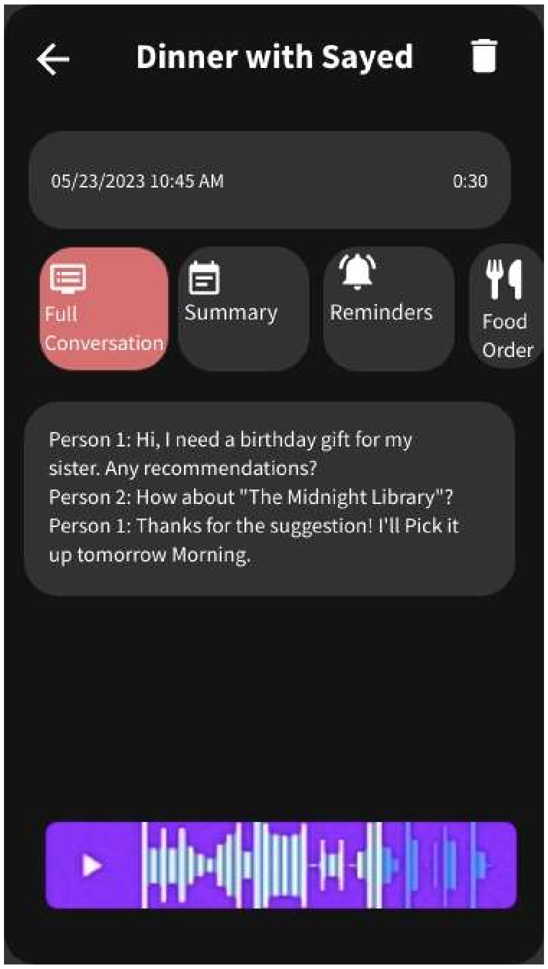
4.4. Screen: Conversation Details

4.4.1. Description

This is the screen the user is redirected to if the user taps on a conversation. On this screen the user can:

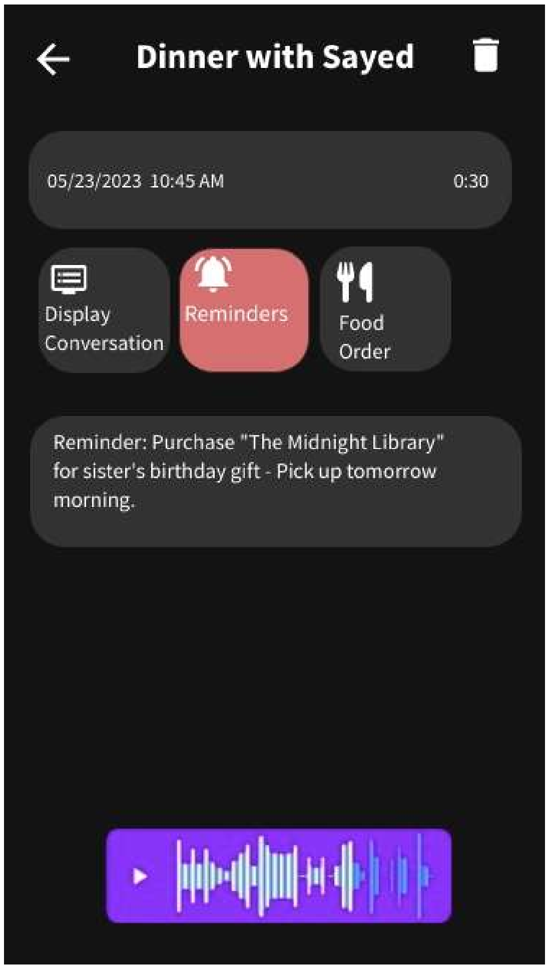
- Edit a conversation
- Delete a conversation.
- Select a process.
- Play a conversation
- View saved process results

4.4.2. Images



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 13: Conversation Details Screen



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

Figure 14: Conversation Details - Reminders Widget Screen

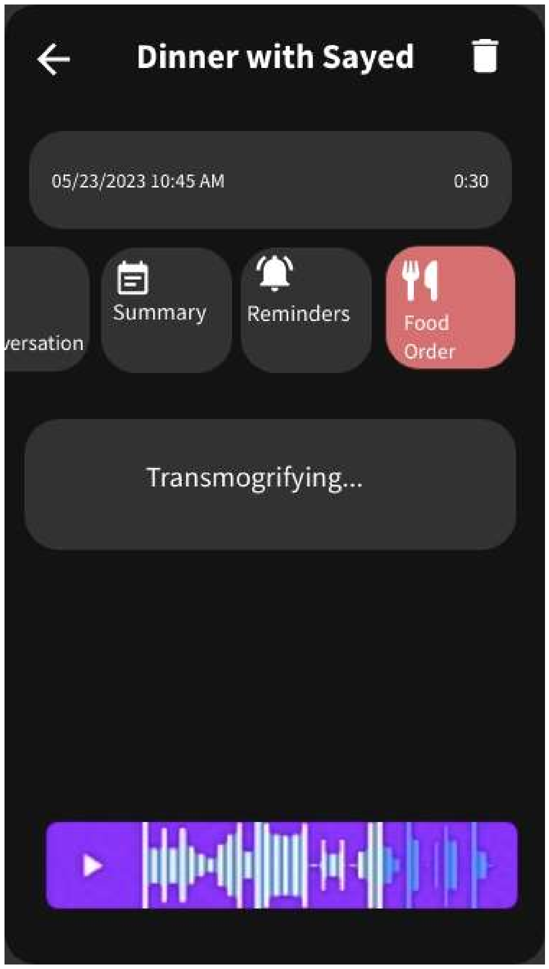


Figure 15: Conversation Details - Missing Result Screen

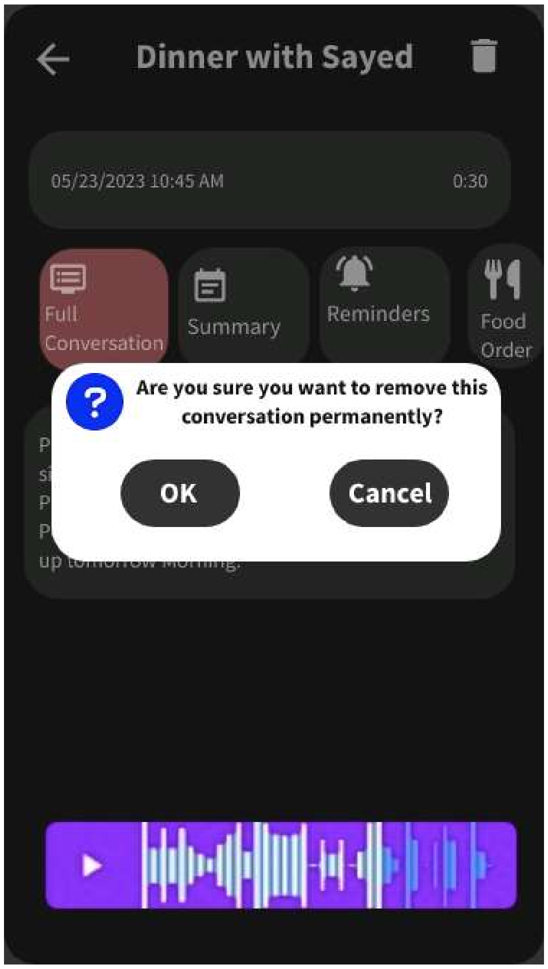


Figure 16: Conversation Details - Delete Prompt Screen

4.4.3. Components

- App Bar
- Icon Button
- Text Button
- Transmog Category Picker
- Transmog List Item
- Text

4.4.4. Internal UI Functionality

- Upon displaying, this screen shall show, from top to bottom:
  - Header
    - Back arrow
    - Conversation Title
    - Delete icon
  - Transmogrifiers in a sideways scrolling list
  - The currently selected transmogrifier's Result widget
  - A Play button

- Tapping on the Delete icon, the app shall show the user a message "Are you sure you wish to remove this conversation permanently?"
    - Tapping "OK," the message shall disappear and the conversation will be deleted.
    - If the user taps "Cancel," the message shall disappear and the conversation is not deleted.
- Tapping the Conversation Title shall put it in edit mode.
    - The text will be highlighted by default
    - The highlighted text shall be deleted if the user begins to type in a new title.
    - If the Conversation Title is in edit mode, then tapping outside of the Conversation Title will save the current text.
- Tapping on any transmogrifier shall display its result widget.
    - If there is no saved result, then the widget will start processing to generate a result.

4.4.5. External UI Functionality

- Tapping on the Back icon displays the Conversation List screen.
- Tapping on the Record button displays the Recording – Started screen.
- If the user taps "OK" to delete the conversation, then display the Conversation List screen.

4.5.Screen: Recording – Started

4.5.1. Description

This screen shall display if the user taps on the Record button on the bottom of the Conversation List screen. On this screen, the app is actively recording the user, and the user can pause or stop a recording.

4.5.2. Images

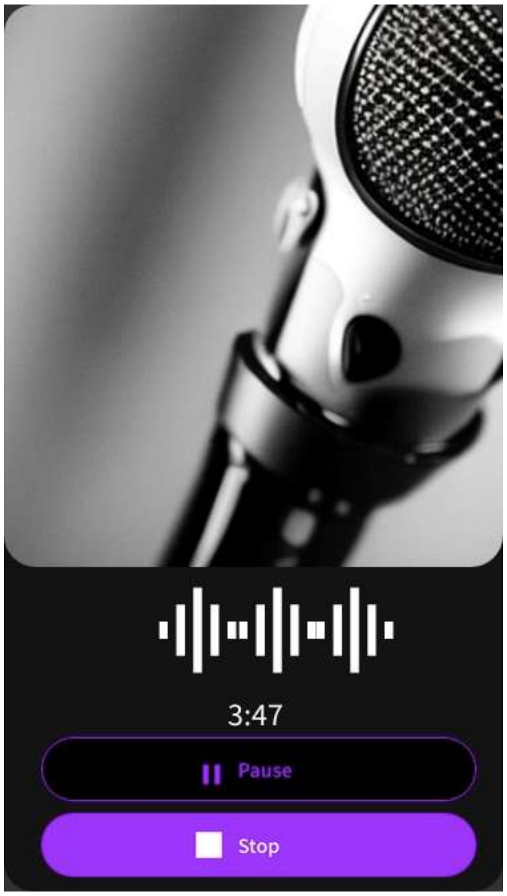


Figure 17: Recording - Started Screen

4.5.3. Components

- Animated Audio Wave
- Image
- Text
- Text Button
- Timer

4.5.4. Internal UI Functionality

- Displaying the screen shall show, from top to bottom and centered horizontally:
    - A static image

    - Animated audio waves
    - A timer that displays minutes and seconds
    - A Pause button
    - A Stop button

- The timer shall update every second with the total duration of the recording.
- The Pause and Stop buttons shall be enabled.
- The animated audio waves shall move.

4.5.5. External UI Functionality

- Tapping the Pause button shall
    - Pause recording audio.
    - Display the Recording –Paused screen.
- Tapping the Stop button shall
    - Stop recording audio.
    - Display the Recording - Stopped screen.
- Tapping Back on the mobile phone's interface shall
    - Display a confirmation message asking the user to confirm cancelling the recording.
    - If the user selects No, then close the confirmation message and cancel the Cancel button action.
    - If the user selects Yes, then display the Conversation List screen and delete any temporary files related to the recording.

4.6.Screen: Recording – Paused

4.6.1. Description

The user will see this screen if they pause the recording process, without stopping it completely. On this screen the user can resume the recording or stop the recording.

4.6.2. Images

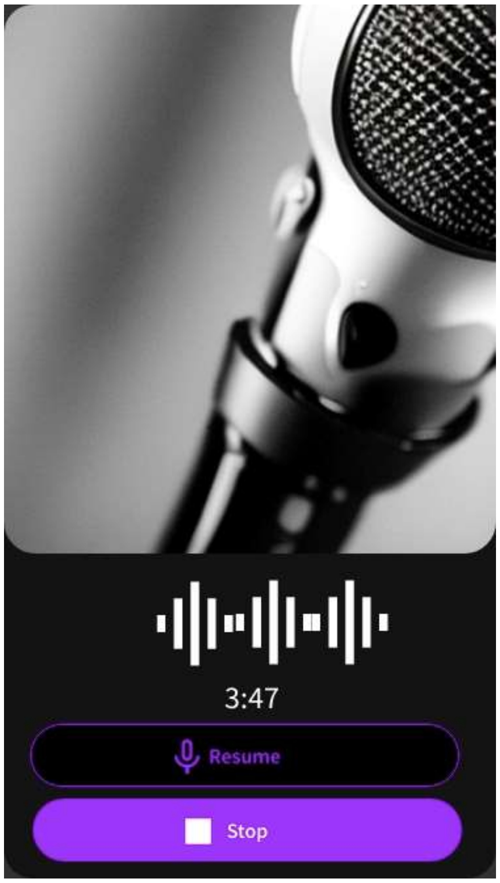


Figure 18: Recording - Paused Screen

4.6.3. Components

- Animated Audio Wave
- Image
- Text
- Text Button
- Timer

4.6.4. Internal UI Functionality

- Upon displaying, the screen shall show, from top to bottom and centered horizontally:
  - A static image
  - Animated audio waves
  - A timer that displays minutes and seconds

  - A Resume button
  - A Stop button

- The timer shall show the total duration of the recording.
- The Resume and Stop buttons shall be enabled.
- The animated audio waves shall not move.

4.6.5. External UI Functionality

- Tapping the Resume button shall
  - Continues recording audio.
  - Displays the Recording –Started screen.
- Tapping the Stop button shall
  - Stops recording audio.
  - Displays the Recording - Stopped screen.
- Tapping Back on the mobile phone’s interface shall
  - Display a confirmation message asking the user to confirm cancelling the recording.
  - If the user selects No, then close the confirmation message and cancel the Cancel button action.
  - If the user selects Yes, then display the Conversation List screen and delete any temporary files related to the recording.

4.7.Screen: Recording – Stopped

4.7.1. Description

This is the screen the user will see if the user stops the recording process. On this screen the user can:

- Save a recording
- Name a recording
- Cancel a recording

4.7.2. Images

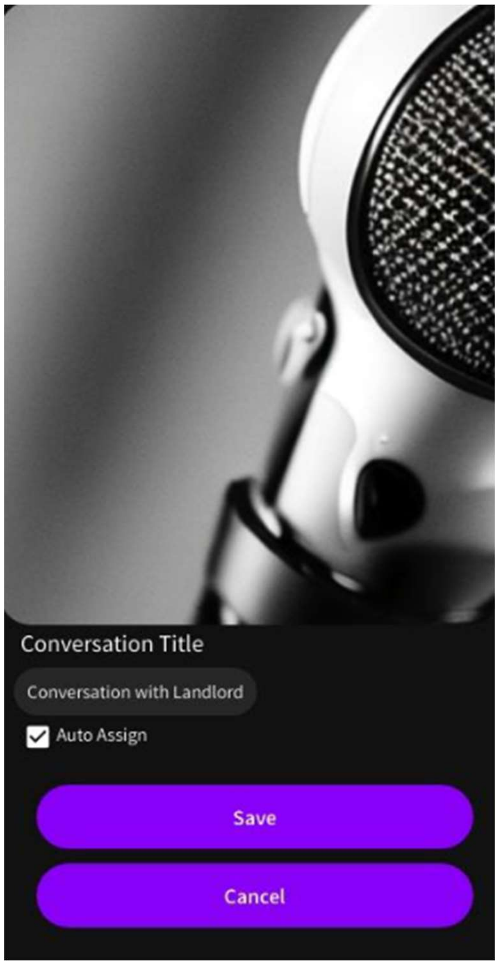


Figure 19: Recording - Stopped Screen

4.7.3. Components

- Checkbox List Tile
- Image
- Text
- Text Button
- Text Field

4.7.4. Internal UI Functionality

- Displaying the screen shall show, from top to bottom:
  - A static image
  - A Conversation Title label
  - A Conversation Title textbox.
  - An Auto-Assign checkbox.
  - A Save button
  - A Cancel button
- The Conversation textbox shall be empty by default.
- The Auto-Assign checkbox shall be checked by default.
- The Save and Cancel buttons shall be enabled by default.
- If the Auto-Assign checkbox is unchecked, then the Conversation textbox shall be enabled and focus will move to it.
- If the Auto-Assign checkbox is checked, then the Conversation textbox shall be disabled and any existing text deleted.
- If the Auto-Assign checkbox is unchecked and the Conversation textbox is empty, then the Save button will be disabled.
- If either the Auto-Assign checkbox is checked or the Conversation textbox is not empty, then the Save button will be enabled.

4.7.5. External UI Functionality

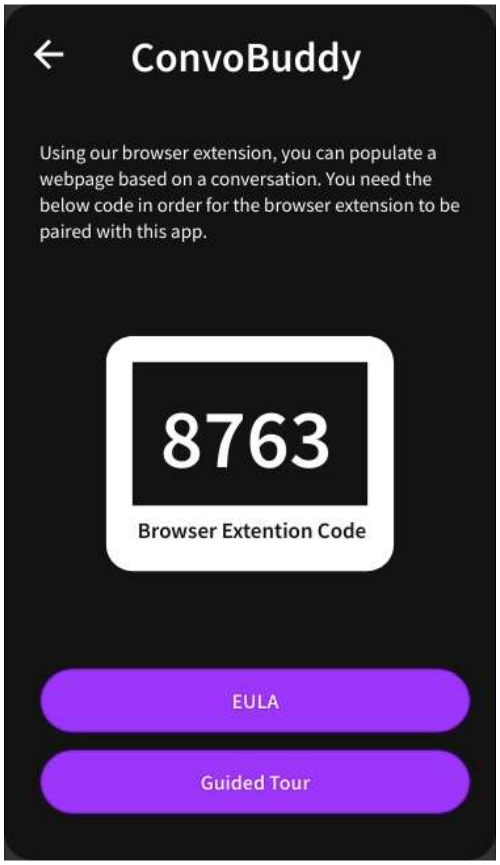
- Tapping the Save button shall
  - Save the audio file to storage.
  - Save the conversation to storage with the value from the Conversation textbox.
  - Start the STT process.
  - If Auto-Assign is checked, then start the transmogrifier process to get a title.
  - Display the Conversation List screen
- Tapping the Cancel button shall
  - Displays a confirmation message asking if the user is sure they want to cancel the save.
  - If the user selects No, then close the confirmation message and cancel the Cancel button action.
  - If the user selects Yes, then display the Conversation List screen and delete any temporary files related to the recording.
- Tapping Back on the mobile phone's interface shall act the same as the Cancel button.

4.8. Screen: Information

4.8.1. Description

If the user taps the Information icon on the top of any screen, the app will display this screen. On this screen, the user can view their Browser Extension code and tap a button to access the Guided Tour.

4.8.2. Images



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.

Figure 20: Information Screen

4.8.3. Components

- App Bar
- Code Viewer Card
- Text
- Text Button

4.8.4. Internal UI Functionality

- Displaying the screen shall show
  - The name of the app at the top.
  - Text that tells users what the browser extension is and why they need the code.
  - A Browser Extension Code label.
  - A Browers Extension Code textbox.

  - A Guided Tour button near the bottom.
  - The Home button at the bottom.

- The Browers Extension Code textbox is disabled and populated with the stored code by default. this shall never change.
- The Guided Tour button is enabled.

4.8.5. External UI Functionality

- Tapping the Guided Tour button shall display the Guide Tour screen.
- Tapping the Home button shall display the Conversation List screen.
- Tapping Back on the mobile phone’s interface shall act the same as the Home button.

4.9.Screen: Guided Tour

4.9.1. Description

This screen shall be displayed to the user if the user taps on the Guided Tour button on the Information screen. This screen shall display a carousel of images that explains how to use and navigate the App correctly.

4.9.2. Images

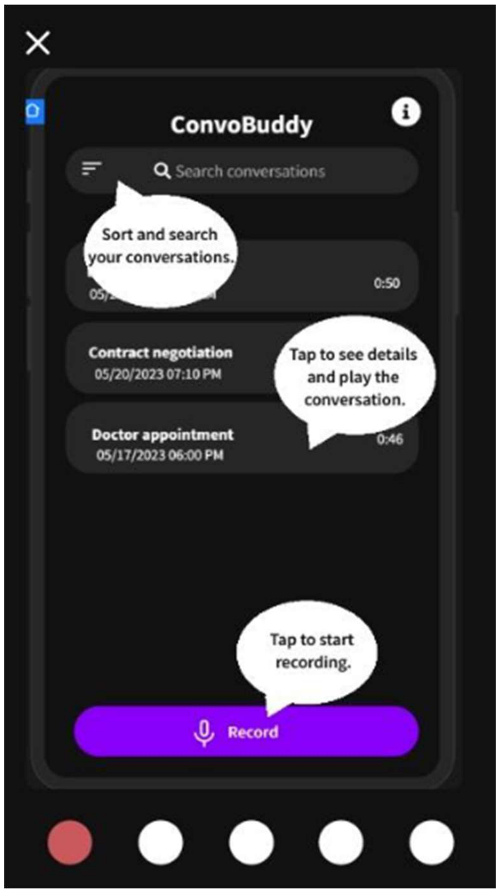


Figure 21: Guided Tour - First Screen

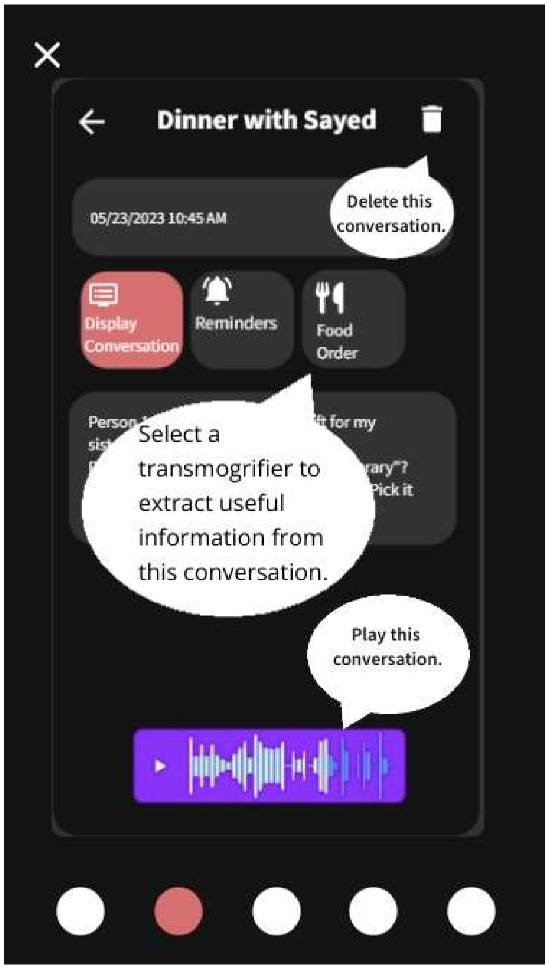

Dinner with Sayed
05/23/2023 10:45 AM
Delete this conversation.
Display Conversation
Reminders
Food Order
Select a transmogrifier to extract useful information from this conversation.
Play this conversation.

Figure 22: Guided Tour - Second Screen

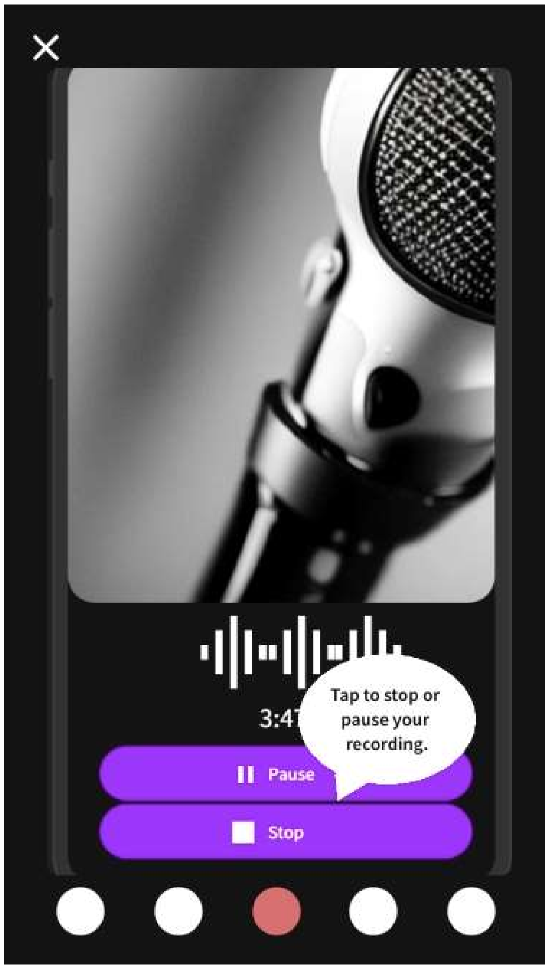


Figure 23: Guided Tour - Third Screen

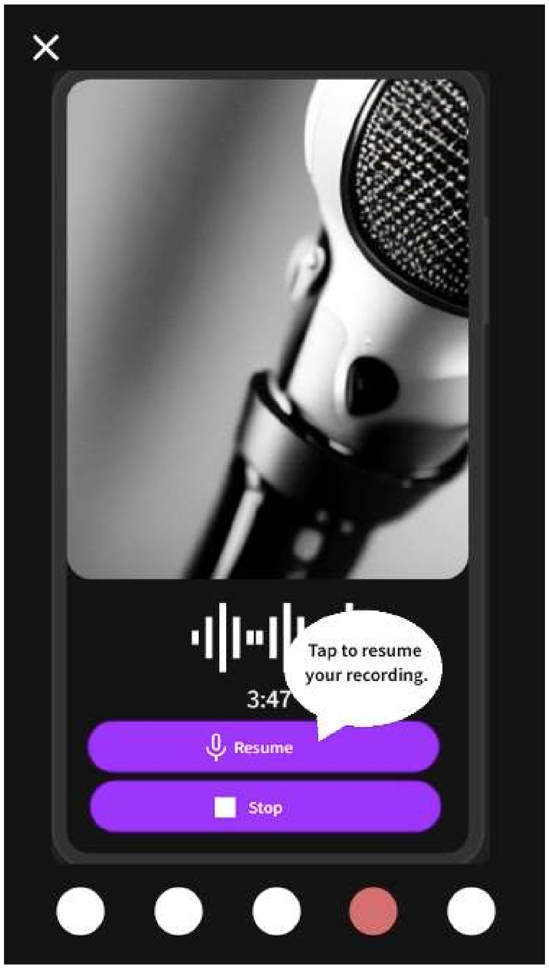


Figure 24: Guided Tour - Fourth Screen

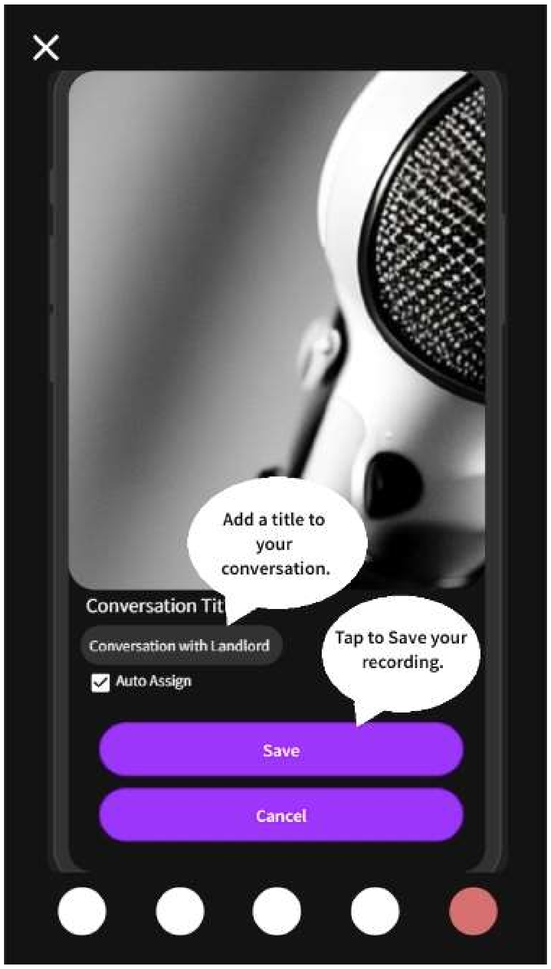

Add a title to your conversation.
Conversation Tit
Conversation with Landlord
Auto Assign
Tap to Save your recording.
Save
Cancel

Figure 25: Guided Tour - Fifth Screen

4.9.3. Components

- Image Carousel

4.9.4. Internal UI Functionality

- Displaying the screen shall show
  - The first image in the image carousel.
  - A Cancel icon in the top left corner.
  - The Home button at the bottom.
- Swiping horizontally on the image carousel shall show the next (left swipe) or prior (right swipe) image.

4.9.5. External UI Functionality

- Tapping the Cancel icon shall display the Information screen.
- Tapping the Home button shall display the Conversation List screen.

4.10. Error Handling

The robust design of ConvoBuddy incorporates stringent error handling to ensure a seamless user experience. Notifications from the form filler browser extension have a 30-second active window, necessitating efficient response times from external dependencies. If a response takes too long, then ConvoBuddy promptly halts the request attempt, thereby maintaining optimal app performance. Instead of leaving users in the dark, the app handles such instances by rendering a meaningful error message for the user. This approach ensures that users are informed about the situation, improving transparency and user experience, even when issues arise.

5. Design Elements for Form-Filler Browser Extension

5.1.Data Design

5.1.1. Summary

The following list of objects are used within the browser extension to send and receive data from BESie. The objects are removed from memory at the earliest possible moment.

5.1.2. Objects

Object Name	**Purpose**
formList	An array that contains all form elements found on the page.
jsonPayload	Holds the JSON representation of the form(s) to be sent to BESie
responseData	The JSON payload that is returned from BESie that contains the form values provided from ChatGPT.

5.1.2.1. Object: formList

```
{
    [           formDomObj1,
     formDomObj2,
     formDomObj3]
}
```

5.1.2.2. Object: jsonPayload

The JSON payload that will be sent to BESie will be structured as the example below.

```
{
 "Form1": {
  "fieldNames": [
   "name1",
   "name2",
   "name3"
  ]
 },
 "form2": {
  "fieldNames": [
   "name1",
   "name2",
   "name3"
  ]
  }
}
```

5.1.2.3. Object: responseData

```
{
 "Form1": {
  "name1": "value1",
  "name2": "value2",
  "name3": "value3"
 },
 "form2": {
  "name1": "value1",
  "name2": "",
  "name3": "value3"
 }
}
```

5.1.3. Source files

Name	Type	Purpose
Index.html	HTML	House the HTML of the pop up
App.js	JavaScript	The main JS file code
Helper.js	JavaScript	Contains helper functions that scrapes the forms and makes WebSocket calls to BESie

5.1.4. External dependencies

Name	Purpose
Bootstrap 5	CSS library that provides styles for layout and UI components such as buttons and text fields.
jQuery	A JavaScript library that allows developers to easily manipulate DOM elements and perform simple animations.

5.2. UX Design

5.2.1. Components

Component	Location	Summary
Enter PIN	Top left-hand corner the screen.	A text box that accepts numeric PIN value.
Fill form button	Beside the “Enter PIN” field to the right.	Scrapes all input fields on the current page and converts them to JSON

5.2.2. Message Catalog

Type	Message ID	Text	Reason
Error	1	Unable to pair with mobile app	Unexpected errors such as loss of internet connection or corrupt data received from BESie.
Error	2	Invalid PIN	Occurs whenever the entered PIN number does not match any mobile app instances.
Error	3	Unable to fill the form	Occurs when the form cannot be filled with the values supplied by BESie. The JSON returned by BESie may be corrupt or not in the correct format.

5.3.Screen: Web browser extension pop up

5.3.1. Description

The main window for the browser extension. A user must enter the Browser Extension Code to start the process for security reasons.

5.3.2. Images

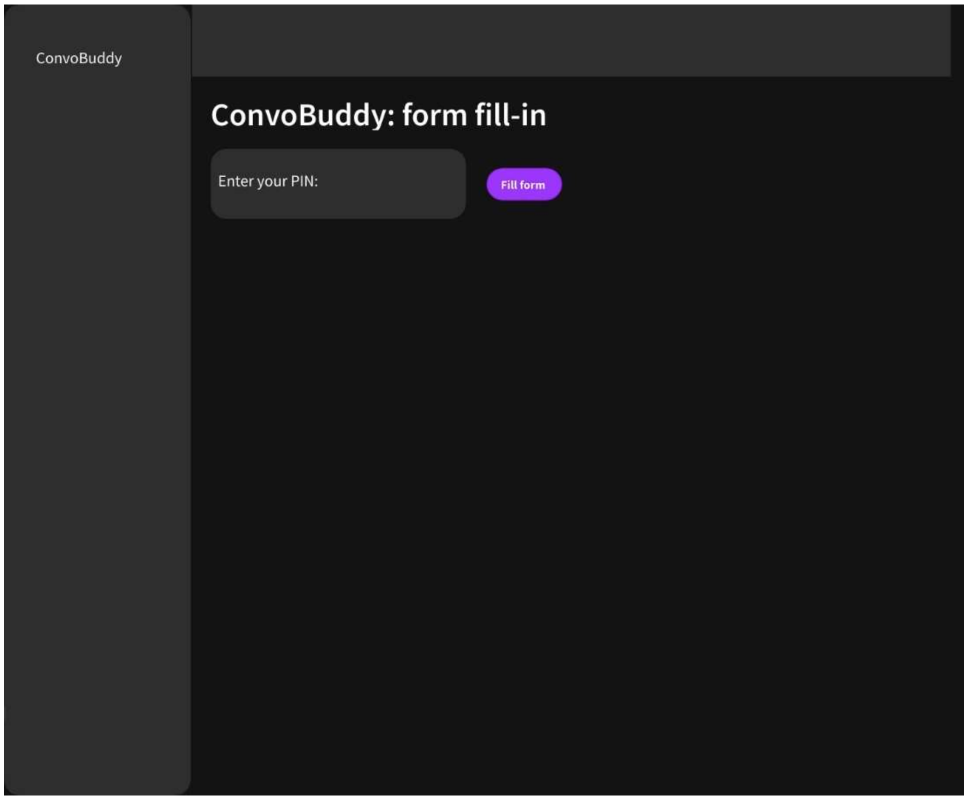


Figure 26: Browser Extension Form Fill-in Request Screen

5.3.3. Components

- PIN text field
- Fill form button

5.3.4. Internal UI Functionality

- The UI shall display a text field that accepts numeric PIN values only and shall be positioned on the upper left-hand corner of the screen.
- The UI shall display a button called "Fill Form" next to the text field.

5.3.5. External UI Functionality

- Upon clicking the "Fill Form" button, the PIN number shall be validated and the PIN, JSON payload containing the field names should be transmitted to BESie. BESie shall return a JSON payload that contains the field values for all forms.

    - After receiving the form values from BESie, the web page shall be populated with those values.

5.4. Error Handling

- If BESie, for any reason, does not provide a value for a specific field, the field will remain blank or populated with any value that was it was populated with beforehand.
- Error messages shall be displayed in a red error box at the top center of the pop up.
- Error message “Unable to pair with mobile app” will be displayed if there is an unexpected error such as loss of an internet connection or any unhandled exception.