

Programmer Guide

(CogniOpen - Nurturing Memory Wellness for Cognitive Impairment)

Submitted to:

University of Maryland Global Campus in partial fulfillment of the course Software Engineering Capstone (SWEN 670)

Submitted By: Team B

Fall Semester 2023

Version 1.0

October 28, 2023

Department of Software Engineering

University of Maryland Global Campus (UMGC)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Issue Date | Changes | Change Made By |
| 1.0 | 09/25/2023 | Initial Version | Team B |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Programmer Guide 7](#_Toc149239610)

[1. Introduction 7](#_Toc149239611)

[**1.1 Purpose** 7](#_Toc149239612)

[**1.2 Intended Audience** 7](#_Toc149239613)

[**1.3 Project Documents** 7](#_Toc149239614)

[**1.4 Acronyms, Definitions, and Abbreviations** 7](#_Toc149239615)

[**1.5 Document Overview** 8](#_Toc149239616)

[**1.6 Document Scope** 9](#_Toc149239617)

[2. Technical Summary 9](#_Toc149239618)

[**2.1 Business Case** 9](#_Toc149239619)

[**2.2 Architecture** 9](#_Toc149239620)

[**2.2.1 CogniOpen Application** 9](#_Toc149239621)

[**2.2.2 External Services** 10](#_Toc149239622)

[**2.2.3 Data Storage** 10](#_Toc149239623)

[**2.3 Use Scenarios** 10](#_Toc149239624)

[**2.4 Licensing** 11](#_Toc149239625)

[3. Technical Rationale 11](#_Toc149239626)

[**3.1 Operating Systems** 11](#_Toc149239627)

[**3.2 Development Languages and Frameworks** 11](#_Toc149239628)

[**3.3 Architecture** 11](#_Toc149239629)

[**3.3.1 CogniOpen Architecture Overview** 11](#_Toc149239630)

[**3.3.2 User Form Input and Validation** 12](#_Toc149239631)

[**3.3.3 Dynamic Environment Configuration** 12](#_Toc149239632)

[**3.3.4 Amazon Transcribe** 12](#_Toc149239633)

[**3.3.5 ChatGPT API** 12](#_Toc149239634)

[**3.3.6 Amazon S3** 12](#_Toc149239635)

[**3.3.7 Design Caveats** 13](#_Toc149239636)

[**3.4 UX Design** 13](#_Toc149239637)

[**3.4.1 User Friendliness** 13](#_Toc149239638)

[**3.4.2 Reintroduction of Functionality** 14](#_Toc149239639)

[**3.5 Security** 14](#_Toc149239640)

[**3.6 Communication Protocols** 15](#_Toc149239641)

[**3.7 Testing Tools** 15](#_Toc149239642)

[**3.7.1 Device Emulator** 15](#_Toc149239643)

[**3.7.2 Physical Device** 15](#_Toc149239644)

[4. Program Code Details 16](#_Toc149239645)

[**4.1 Code Structure** 16](#_Toc149239646)

[**4.1.1 Main Application Folder** 16](#_Toc149239647)

[**4.1.2 Assets** 16](#_Toc149239648)

[**4.1.3 Tests** 18](#_Toc149239649)

[**4.1.4 CogniOpen Library Code** 18](#_Toc149239650)

[**4.1.4.1 Backend Source Code** 19](#_Toc149239651)

[**4.1.4.2 UI Source Code** 20](#_Toc149239652)

[**4.2 Dynamic Environment Configuration File** 21](#_Toc149239653)

[**4.3 Dependency Enumeration** 22](#_Toc149239654)

[**4.4 AWS User Permissions** 23](#_Toc149239655)

[**4.5 Known Issues** 23](#_Toc149239656)

[5. Appendix 24](#_Toc149239657)

[**5.1 Source Control** 24](#_Toc149239658)

**Table of Tables**

[Table 1: Project Documents 7](#_Toc148966428)

[Table 2: Definitions, Acronyms, and Abbreviations 8](#_Toc148966429)

[Table 3: Environment File Details 22](#_Toc148966430)

[Table 4: Dependency Descriptions 23](#_Toc148966431)

**Table of Figures**

[Figure 1: Main Application Folder 16](#_Toc148966432)

[Figure 2: Asset Folder 17](#_Toc148966433)

[Figure 3: Test Directory 18](#_Toc148966434)

[Figure 4: Library Code 19](#_Toc148966435)

[Figure 5: Backend Source Code 20](#_Toc148966436)

[Figure 6: UI Source Code 21](#_Toc148966437)

[Figure 7: Environment File 22](#_Toc148966438)

[Figure 8: Branching Structure 24](#_Toc148966439)

# Programmer Guide

# 1. Introduction

## **1.1 Purpose**

CogniOpen is an application that helps people with cognitive impairments maneuver their day-to-day lives. The application aims to support people with such impairments by providing modern technical solutions to daily problems. With the use of artificial intelligence (AI), third-party tools, and a user-friendly design, people with cognitive impairments and their caregivers can remember critical information about their daily lives. The success this application lies in the technical solutions our team and our accompanying team are providing and the blueprint provided for future implementers.

## **1.2 Intended Audience**

This guide is intended for programmers who want to understand the technical details of the application, including the architecture, use scenarios, code structure, algorithms, tests, and known issues. This guide may be used by current Team B developers of the CogniOpen application and any future implementers of the CogniOpen application capabilities.

## **1.3 Project Documents**

This document is one part of the deliverables for the CogniOpen project. The other documents are listed in the table below with the most recent version of each at the time of editing this Programmer Guide.

|  |  |  |
| --- | --- | --- |
| Document | Version | Date |
| Project Plan (PPL) | 1.2 | 10/28/2023 |
| Software Requirements Specification (SRS) | 1.2 | 10/28/2023 |
| Technical Design Document (TDD) | 1.1 | 10/28/2023 |
| Test Plan (TP) | 1.1 | 10/28/2023 |
| Programmer Guide (PG) | 1.0 | 10/28/2023 |
| Deployment and Operations Guide (DOG) | 1.0 | 10/28/2023 |
| Software Test Report (STR) | - | - |
| User Guide (UG) | - | - |
| Traceability Matrix (TM) | - | - |

Table 1: Project Documents

## **1.4 Acronyms, Definitions, and Abbreviation**

|  |  |
| --- | --- |
| Term | Definition |
| ADO | Azure DevOps |
| API | Application Programming Interface |
| App | A program that is included on the User’s mobile device |
| AWS | Amazon Web Services |
| HIPPA | Health Insurance Portability and Accountability Act |
| HTTPS | Hypertext Transfer Protocol Secure |
| IAM | Identity and Access Management |
| IDE | Integrated development environment |
| iOS | iPhone Operating System |
| Mobile Device | A smart phone, tablet, or some other portable computer with either the iOS or Android operating system |
| MVC | Model View Controller |
| OS | Operating System |
| REST | Representational State Transfer |
| SDK | Software Development Kit |
| SNS | Simple Notification Service |
| TDD | Technical Design Document |
| UI | User Interface |
| UMGC | University of Maryland Global Campus |
| UX | User Experience |

Table 2: Definitions, Acronyms, and Abbreviations

## **1.5 Document Overview**

This document is organized into five logical sections – an introduction, a technical summary, a development rationale, detailed programmatic information, and an appendix. These five sections will work to provide technical responses to challenges the CogniOpen Team B encountered while developing their features. The introduction section will provide an overview of the document, the purpose for writing it, the intended audience, and high-level summarizations. The technical summary, development rationale, and detailed programmatic information sections will provide highly technical information about specific code implementations. The purpose of these three sections is to remove any ambiguity for future implementers. Finally, the appendix shall provide supporting information and any additional items that may provide reference to the previous technical sections.

## **1.6 Document Scope**

This document will only cover the technical components associated to features assigned to Team B. All discussions surrounding the features and components assigned to the accompanying team will be detailed in their Programmer Guide.

# 2. Technical Summary

## **2.1 Business Case**

The customer, Dr. Mir Assadullah, has asked Team B and our accompanying team to provide a technical solution to assist individuals with cognitive impairments. As more individuals are impacted by cognitive impairments, there is an increasing need for technical solutions that aid these people. As a result, the customer has asked our two teams to develop one mobile application. The following sections will detail the architecture, requirements, and use scenarios provided to our teams.

## **2.2 Architecture**

There are three components that make up the CogniOpen architecture: the CogniOpen application, external services, and data storage. These three components will be described in more detail in the following sections. Together, they will provide the user with the front-end UI and business logic, the third-party utilities that perform critical system functionalities, and the services that host data needed by the application, in that order. Without any one of these components, the CogniOpen application would not be successful.

### **2.2.1 CogniOpen Application**

The CogniOpen application consists of six main features assigned to Team B: recording audio, providing a virtual assistant, user login, user registration, profile management, and a guided tour of the application. Each feature will be outlined in detail in a subsequent section. The recording audio feature allows users to record conversations. The CogniOpen application will take these conversations, store the information, and convert the audio content to text for further analysis. The virtual assistant allows users to query ChatGPT for general questions or contextual analysis of the conversations they have had. User login provides a biometrically secured login to the application, ensuring unique credentials and non-repudiation. If a user has not already created an account with CogniOpen, they can utilize the user registration feature to create a new account. After a new account has been created, the profile management feature allows users to modify information associated to their account. Finally, the CogniOpen application will have a guided tour that users can repeatedly activate to learn or remember how to navigate the application. These features will provide a robust application that has a simple and user-friendly interface.

### **2.2.2 External Services**

External services are third-party utilities that provide critical business value to the CogniOpen development team. There are two external services that will be utilized by the CogniOpen application, including Amazon’s Transcribe and OpenAI’s ChatGPT. Amazon Transcribe can take audio content and automatically convert it to text. After a video or audio recording is completed, the audio portion of either instance will be sent to Amazon Transcribe, which will return a raw dump of the audio provided in text format. ChatGPT provides the CogniOpen application AI textual analysis. Team B will utilize ChatGPT to query conversations. The application will provide ChatGPT with the text retrieved from Amazon Transcribe and a query. In response, ChatGPT will provide a human-like response to the query.

### **2.2.3 Data Storage**

Data storage is critical for applications to succeed. The CogniOpen application will utilize two sources of data storage: cloud storage for physical files, and database storage for user information. For cloud storage, Amazon S3 has been chosen to provide a repository of video and audio files that can be accessed by the application. However, due to some strict pricing restrictions, there may be a limit to what individuals can store in Amazon S3. Additionally, to store user information, the CogniOpen application will use sqflite, a SQLite plugin for Flutter. Sqflite will provide SQL syntactical support while integrating natively into the architecture. Team B is using this database technology to store user information for their profile and the audio recordings that users will save.

## **2.3 Use Scenarios**

The use scenarios used to design the application were provided by the customer. Below is a list of use scenarios given to the team:

* Parse audio recordings and save the text for future analysis.
* Check if the user agreed to a meeting or appointment.
* Repeatable user guidance on using the application.

Additionally, as development continues there are several additional use scenarios the application could support. Below is a list of use scenarios Team B has come up with:

* Students or professors recording lectures for playback and contextual analysis.
* Lawyers or police officers recording interviews.
* Therapists who want to provide more in-depth treatment plans.

## **2.4 Licensing**

The University of Maryland Global Campus (UMGC) provided the guidance and oversight for the development of the CogniOpen application. Thus, the CogniOpen application is completely free and open source for future developers or implementers to take the code and make improvements. However, due to the sensitive nature of the data that may be stored by the application, users will be required to assert that they are aware of the risks they accept by using the application.

# 3. Technical Rationale

## **3.1 Operating Systems**

The CogniOpen is being designed as a mobile application. Both iPhone Operating System (iOS) and Android operating systems (OS) will be supported for the deployment and operation of the application. These two OSs were chosen due to their widespread popularity and adoption. Additionally, the programming languages used for development provide simple and native support for these two OSs, minimizing the development work for this effort.

## **3.2 Development Languages and Frameworks**

Dart is the primary programming language will be used to develop the CogniOpen application. The CogniOpen development team is using the Dart software development kit (SDK) version 3.1.3, the most up to date version at the time of development. Dart, an open-source programming language managed by Google, provides the team with seamless mobile application integration with Flutter. Flutter is another open-source technology that provides native support for mobile applications and numerous third-party library integrations. The development team is using version 3.13.6 of Flutter during the time of initial implementation.

## **3.3 Architecture**

### **3.3.1 CogniOpen Architecture Overview**

The CogniOpen application follows a model-view-controller (MVC) design pattern. The data storage component of the application, detailed above, provides the model for the data presentation. This gives developers a blueprint for how to present information to the users in the view. The view is how the end users will interact with the application. Finally, all the business logic that handles how the application functions is taken care of in the controller. Each of these components is developed in a modular class-based infrastructure. This allows for rapid development, simple refactor, and minimal code conflicts.

### **3.3.2 User Form Input and Validation**

User input is only taken in three parts of the system: account creation and registration, profile management, and the virtual assistant. For each of these components, all user input will be taken in and sanitized to the furthest extent possible. Additionally, for phone numbers and email addresses shall be parsed to ensure that they are in the proper format.

### **3.3.3 Dynamic Environment Configuration**

To provide dynamic environment configuration, the CogniOpen application requires an environment file that will be loaded at runtime. This configuration file will be used to store access data for the various third-party tools that are being integrated. The following sections will include the specific environment configurations required for those components.

### **3.3.4 Amazon Transcribe**

Amazon Transcribe is the third-party utility the CogniOpen development team is using to convert audio content to text. The team is using the Dart package ‘aws\_transcribe\_api’ version 2.0.0 or higher to accomplish this task. It requires an account to be created with Amazon Transcribe and the user information must be entered into the application’s environment file. Additionally, for this feature to work, the application must be connected to the internet, and the audio file must first be uploaded to an Amazon S3 Bucket before a transcription job can be started.

### **3.3.5 ChatGPT API**

ChatGPT is the third-party utility the CogniOpen development team is using to provide textual analysis and virtual assistance. The team is using the Dart package ‘dart\_openai’ version 3.0.0 or higher to interface with the ChatGPT application programming interface (API). Using ths API requires registering for an OpenAI account and creating an API key. The key must then be entered into the application’s environment file. Additionally, for this feature to work, the application must be connected to the internet.

### **3.3.6 Amazon S3**

Amazon S3 is the third-party utility the CogniOpen development team is using to provide storage for audio recording files. Amazon Transcribe requires the audio files to be in an Amazon S3 Bucket before starting a transcription job. The team is using the Dart packages ‘aws\_sns\_api’ and ‘aws\_s3\_api,’ both version 2.0.0 or higher. This requires an account to be created with AWS and the user information must be entered into the application’s environment file. Each developer will need to create a unique bucket and queue name for their instance to interface with. This resolves the issue with globally unique bucket names not colliding with other people’s information. Additionally, for this feature to work, the application must be connected to the internet.

### **3.3.7 Design Caveats**

While developing these features for the CogniOpen application, several design decisions were made that may not be in line with industry best practices. Ideally, a single company AWS account and OpenAI account would be used to interface with those services. An enterprise-style license would ensure that there would be limited overhead for developer environment setup. Additionally, S3 storage is very limited for free accounts. Rigorous test scenarios cannot be exercised due to read and write limitations imposed on the development team by AWS.

## **3.4 UX Design**

The user experience (UX) design for the CogniOpen application is a crucial aspect of the product's development. The team has focused on accomplishing two main goals with the UX design - providing a user-friendly interface and reintroducing the features and functions of the application in case users forget. These two facets of UX design are deeply ingrained within the team's development philosophies, as they strive to always provide the best possible product to their users.

### **3.4.1 User Friendliness**

User friendliness is crucial for the success of the CogniOpen application, especially when dealing with individuals with cognitive impairment. The application's ease of use and friendliness can either enable the users to achieve their objectives or discourage them from using the application further. To address this, the CogniOpen team has implemented several navigation features that make the interface of the application intuitive and easy to use. The application is designed in such a way that users can easily find what they are looking for without getting lost.

Navigation is a critical aspect of the CogniOpen UX. It enables users to move between different sections and features of the application, allowing them to quickly find what they are looking for. Below are the key parts and tools that facilitate clear navigation throughout the application:

1. Upper Static Menu: The upper static menu is a fixed menu bar that remains visible at the top of the screen, regardless of the user's location within the application. It contains buttons to the most frequently used sections of the application –the back button and menu button.
2. Lower Static Menu: Similar to the upper static menu, the lower static menu is a fixed menu bar that remains visible at the bottom of the screen. It contains buttons to the most frequently used sections of the application – the home button, gallery button and virtual assistance button.
3. Kebab Menu List: The kebab menu is a three-dot icon that appears next to items in a list, indicating that additional options are available. When users click on the kebab menu, a list of actions related to the item appears. This type of menu allows you to have an indefinite number of elements without cluttering the initial experience.
4. Back Button: The back button is a navigation element that allows users to return to the previous screen or step in the application. The button will be located statically at the top of the screen.
5. Floating Buttons: The microphone button and other floating buttons offer users easy access to frequently used actions, which can be quite useful while interacting with the virtual assistant. The microphone button is particularly convenient as it streamlines the UX, making it smoother and more intuitive.
6. Human Messenger Design: The human messenger design is a user interface element that simulates a conversation with virtual assistance. The message screen will simulate a human-like text message screen to give the user a more human experience when communicating with the virtual assistance.

### **3.4.2 Reintroduction of Functionality**

As users with cognitive impairments load the application repeatedly, the likelihood that they will forget how to fully utilize the application is high. This can lead to frustration and reduced engagement with the application. To address this issue, the CogniOpen development team has decided that providing a streamlined way of reintroducing the system’s functionality should be on the top of the priority list for UX design and development. To achieve this goal, the CogniOpen team has implemented the reintroduction of functionality in various ways. For instance, the team has created a series of onboarding screens that appear when the user first logs in. These screens provide a brief overview of the application’s core functionality, making it easier for the user to recall how to use it. Additionally, the team has created a guided tour mode that can be accessed at any time from within the application. This mode provides step-by-step guidance on how to perform specific tasks within the application, allowing users to refresh their memory and learn new features.

## **3.5 Security**

The CogniOpen application will take a defense in depth approach to security. Providing biometric authentication for the users will ensure that password complexity and security will not concern users. Biometric authentication guarantees that only that user can access their information. Additionally, for all local storage, the CogniOpen application will natively inherit the host device’s security protocols. Android and iOS devices provide strong local data protection, and the CogniOpen application will take advantage of that. Finally, the application will ensure that strict Amazon Web Services (AWS) security protocols are in place to ensure that user data, including audio and video recordings, are secured from outside access. This will include strict identity and access management (IAM) rules, minimal read and write access, and data object integrity. The combination of these security measures will provide users with the comfort that their data is being stored in a safe, but accessible, manner.

## **3.6 Communication Protocols**

All communication within and outside of the CogniOpen application will use HTTPS. This will enforce secure, encrypted, and reliable transmission of communications between the user, the CogniOpen application, and the third-party tools that the application will utilize. Additionally, there will be a representational state transfer (REST) API that the CogniOpen application will use when communication with AWS. This REST API will ride on top of HTTPS but provides an independently implementable protocol.

## **3.7 Testing Tools**

### **3.7.1 Device Emulator**

For testing, device emulation was the preferred platform. The team utilized emulated devices in order to test a wider array of environments, ensuring that the software is platform agnostic and portable. Emulation also allowed for a low barrier of entry for rapid testing. Android devices are the preferred test platform, due to the integration with the Android Studio integrated development environment (IDE). Specifically, the team used Google Pixel 7 emulated devices for the majority of testing procedures. However, the team did encounter a few problems when testing on emulated devices. Chief among them was the verification that features were working. For example, when testing the microphone and audio components, a critical part in testing audio recording, the emulated device added confusion and complexity. When using an emulated device, ensure that it is using the host microphone and ensure that the volume is turned all the way up.

### **3.7.2 Physical Device**

Physical device testing was utilized sparingly in our test process. Certain tests were made significantly easier or necessitated the use of a physical device. Early in development, the team was struggling with biometric verification to assist users with logging in and utilized a physical device which validated the code in test. Without the use of the physical testing device, biometric authentication would have been removed or went unverified. Future implementers may want to experiment with getting biometric authentication working through the emulator, but the development team decided to utilize physical devices for these purposes.

# 4. Program Code Details

## **4.1 Code Structure**

### **4.1.1 Main Application Folder**

The main application folder, cogniopenapp, is broken up into the following components:

* README.md – A markdown file containing critical environment setup information including AWS setup, OpenAI account setup, configuring the application, and running the software.
* assets/ – The assets folder contains images, icons, seed data, and testing artifacts needed to run, build, and test the application.
* lib/ – The lib folder contains all of the software that makes up the CogniOpen application.
* pubspec.yaml – This configuration file specifies all of the dependencies and versions of the dependencies that the application needs. Additionally, the assets that the application will load and some other metadata may be included in this configuration file.
* test/ – The test folder contains all of the unit tests written to automate the testing of the CogniOpen application.

A screen shot of a computer program

Description automatically generated

Figure 1: Main Application Folder

### **4.1.2 Assets**

The assets folder contains four sub-folders: icons, images, seed\_data\_files, and test\_images. The icons folder contains all the images that are used within the application to ensure the UI works and is user friendly. The images folder contains the images that are embedded into different components of the UI, including the background and guided tour images. The seed\_data\_files are files that are added to the database in order to rapidly populate information for testing purposes. Finally, the test\_images folder is used to populate the application with test images to use within the accompanying team’s applications.

A screenshot of a computer program

Description automatically generated

Figure 2: Asset Folder

### **4.1.3 Tests**

The tests folder contains all of the unit tests for the application. Each file inside the folder has a corresponding file in the lib folder, referenced below. Each test file is created with the following format: <feature>\_test.dart. If a unit test needs mocked up content, those mocks are stored in the mocks folder.

A screen shot of a computer

Description automatically generated

Figure 3: Test Directory

### **4.1.4 CogniOpen Library Code**

The CogniOpen library code is broken into the following components:

* main.dart – This is the initial file loaded at application start time. This file is responsible for loading the proper initial functional UI components and the environment file.
* Src – The src folder is where all of the CogniOpen backend source code is stored. This includes database interactions, AWS interfaces, location services, and more.
* ui – The ui folder is where the CogniOpen UI code is stored. This includes every functional UI component that is included as part of the application deployment.

A screen shot of a computer code

Description automatically generated

Figure 4: Library Code

### **4.1.4.1 Backend Source Code**

The backend source code folder is the definitive source for all backend manipulation content within the CogniOpen application. For the Team B features, the team is utilizing the following files:

* address.dart
* conversation.dart
* database/app\_database.dart
* database/app\_database\_seed\_data.dart
* media
* media\_controller
* s3\_connection
* utils/constants.dart
* utils/directory\_manager.dart
* utils/file\_manager.dart

A screenshot of a computer program

Description automatically generated

Figure 5: Backend Source Code

### **4.1.4.2 UI Source Code**

The UI source code folder is the definitive source for all front-end, user-facing, content within the CogniOpen application. For the Team B features, the team is utilizing the following files:

* assistantScreen.dart
* audioScreen.dart
* homeScreen.dart
* profileScreen.dart
* registrationScreen.dart

A screen shot of a computer

Description automatically generated

Figure 6: UI Source Code

## **4.2 Dynamic Environment Configuration File**

The dynamic environment configuration file is a vital component to successful application execution. The specifics around why the development team chose to go with this approach is outlined above. This section will breakdown the components of the environment file. The file, located in the main application folder, and named .env, contains the following information.

|  |  |
| --- | --- |
| Key Value | Description |
| accessKey | Each user’s specific access key for their AWS user account. |
| secretKey | Each user’s specific secret key for their AWS user account. |
| region | The region where the S3 bucket will be located. |
| snsTopicName | The AWS Simple Notification Service (SNS) topic name, used for sending and receiving messages from AWS. |
| videoS3Bucket | The name of the bucket that will store all audio, video, and image files. |
| OPEN\_AI\_API\_KEY | Each user’s specific OpenAI API key for their OpenAI account. |

Table 3: Environment File Details

A computer screen shot of a program

Description automatically generated

Figure 7: Environment File

## **4.3 Dependency Enumeration**

The following table will only include dependencies used by the CogniOpen Team B services. While there may be additional dependencies, the accompanying team will provide them.

|  |  |  |
| --- | --- | --- |
| Dependency Name | Version | Description |
| avatar\_glow | 2.0.2 | Provides a background glow to widgets |
| aws\_client | 0.4.1 | Provides libraries for accessing generic AWS capabilities |
| aws\_s3\_api | 2.0.0 | Provides libraries for accessing AWS S3 storage |
| aws\_transcribe\_api | 2.0.0 | Provides libraries for accessing AWS Transcribe |
| cupertino\_icons | 1.0.2 | Icons used by the Cupertino widgets within Flutter |
| dart\_openai | 3.0.0 | Provides libraries for accessing ChatGPT |
| flutter | Latest | Mobile application support for Dart |
| flutter\_dotenv | 5.1.0 | Provides the ability to use dynamic environment configuration files |
| flutter\_sound | 9.2.13 | Play and record sound |
| flutter\_tts | 3.8.3 | Convert text to speech |
| geocoding | 2.1.1 | Provides location encoding support |
| geolocator | 10.1.0 | Provides location-specific support |
| http | 0.13.6 | Allows flutter applications to make HTTP requests |
| local\_auth | 2.1.7 | Integrates native biometric authentication capabilities into the flutter application |
| mockito | 5.4.2 | Provides the ability to mock features or APIs for testings |
| path | 1.8.3 | Allows the application manipulate paths |
| path\_provider | 2.1.1 | Provides the application access to the local subsystem |
| permission\_handler | 11.0.1 | Allows the application to check and verify permissions |
| plugin\_platform\_interface | 2.1.6 | Provides flutter plugin interfaces natively |
| speech\_to\_text | 6.3.0 | Convert speech to text |
| sqflite | 2.3.0 | Provides the application database support |

Table 4: Dependency Descriptions

## **4.4 AWS User Permissions**

For interacting with AWS services, Transcribe and S3, for managing audio recording, the following permissions need to be added to each user’s account:

* AdministratorAccess-Amplify
* AmazonS3FullAccess
* AmazonTranscribeFullAccess

Without these permissions, audio files will not be able to be stored in the S3 bucket and Transcribe will not be able to perform the transcription services.

## **4.5 Known Issues**

During development, issues are bound to happen. This section will describe all known issues, but there may be additional issues the team is not currently aware of.

**AWS Region Restriction:** At this time, the CogniOpen application cannot support any S3 regions other than us-east-1. In order for each application instance to share common code, this AWS region must be used. This helps reduce overhead and increase development velocity.

**AWS Transcribe Limits:** The AWS Transcribe free tier has a hard limitation on transcription minutes. Each account is limited to 60 minutes of transcription services per month. The CogniOpen development team minimizes this risk by having each developer create their own free tier account, but an active developer or tester may run into this restriction quickly.

**ChatGPT Limits:** There are inconsistent outcomes when using ChatGPT to process large contents of data through the API. For long audio recordings, ChatGPT may not be able to appropriately process the text.

# 5. Appendix

## **5.1 Source Control**

For source control and code management, the CogniOpen development team is using GitHub as the code repository. The code can be found at the following location: https://github.com/umgc/fall2023. This code is available to the public but would require reaching out to the customer to be able to contribute to it directly.

To ensure that the two teams were able to be kept in line with each other, while not squashing each other’s changes, the team adopted the following branching structure to minimize code conflicts:

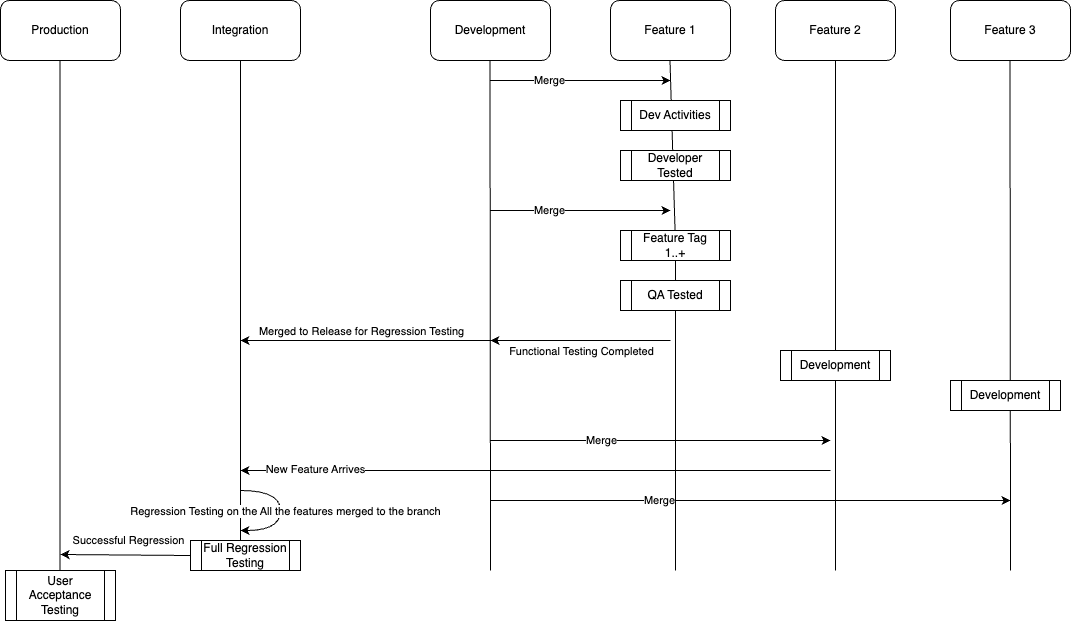


Figure 8: Branching Structure

This branching structure was followed by both teams and has led to a harmonious development effort. Finally, we installed a process to ensure that both teams had visibility into all features that were being developed. This required that one person from each team perform a review on the code prior to merging. This ensured that there was favoritism, and each team had a voice.