

# Programmer's Guide

Harkify Mobile Application

Milestone 3 – Bravo Team

Software Engineering Project

SWEN 670

July 28, 2021

Version 1.1

## Revision History

| Date       | Version | Description                          | Author     |
|------------|---------|--------------------------------------|------------|
| 07/19/2021 | 1.0     | Initial Release                      | Team Bravo |
| 07/28/2021 | 1.1     | Revision to Section 4, other editing | Team Bravo |
|            |         |                                      |            |

## Contents

|  |    |
|--|----|
| 1 Introduction .....                               | 1  |
| 1.1 Purpose .....                                  | 1  |
| 1.2 Intended Audience .....                        | 1  |
| 1.3 Technical Project Stakeholders .....           | 1  |
| 1.4 Definitions, acronyms, and abbreviations ..... | 2  |
| 1.5 References .....                               | 2  |
| 2.0 System Architecture .....                      | 3  |
| 2.1 Architectural Overview .....                   | 3  |
| 2.2 Architectural Design .....                     | 4  |
| 3.0 Application Setup .....                        | 5  |
| 3.1 Git and Code Deployment .....                  | 5  |
| 3.1.1 Download Git .....                           | 5  |
| 3.1.2 Cloning Harkify Repository .....             | 6  |
| 3.1.3 Install Flutter SDK .....                    | 7  |
| 3.1.4 Install Java 8 SDK .....                     | 7  |
| 3.1.5 Install Android Studio .....                 | 8  |
| 3.1.6 Install Visual Studio Code .....             | 9  |
| 3.1.7 Flutter Doctor Checks .....                  | 10 |
| 3.1.8 Test Out Dependencies .....                  | 10 |
| 3.2 Picovoice Overview .....                       | 13 |
| 3.2.1 Wake Words .....                             | 14 |
| 3.2.2 Inferences .....                             | 14 |
| 3.2.3 Training .....                               | 15 |
| 3.2.4 Integration into Harkify .....               | 15 |
| 3.3 Azure Speaker Recognition Overview .....       | 15 |
| 3.4 Code Structure .....                           | 16 |
| 3.4.1 Source Code .....                            | 16 |
| 3.5 Data Security .....                            | 17 |
| 4.0 Running the Application .....                  | 17 |
| 4.1 Installing Git .....                           | 17 |
| 4.2 Installing Flutter and VS Code .....           | 17 |
| 4.3 Installing Android SDK .....                   | 18 |
| 4.4 Voice Recognition Services .....               | 18 |

|                                      |    |
|--------------------------------------|----|
| 4.5 Harkify Application.....         | 18 |
| 5.0 Challenges/Issues/Concerns ..... | 20 |
| 6.0 Additional Help .....            | 20 |
| 7.0 License Information.....         | 21 |

# 1 Introduction

## 1.1 Purpose

The Programmer's Guide describes how to develop and maintain the Harkify application with detailed descriptions of how the application was developed for future developers. For more information on installing and maintaining the user environment, refer to the Deployment and Operations Guide. In this document the developers can see the various tools and services utilized, the language selected, and the application's architecture overall.

## 1.2 Intended Audience

This manual is intended for programmers who are interested in developing and maintaining the Harkify applications. This manual assumes that the programmers have no previous knowledge of the Harkify application, and the tools and services used for its development. However, it does assume that the programmers have access to this Programmer's Guide and the Deployment and Operations Guide documentation, ability to understand and read Dart programming language in Flutter that was used to develop the application and supplements the limited Flutter information in this manual.

## 1.3 Technical Project Stakeholders

The project stakeholders for the Harkify application are listed below in table 1.

| Name               | Role                        |
|--------------------|-----------------------------|
| Dr. Mir Assadullah | Stakeholder (Project Owner) |
| Raul Benavides     | Project Manager             |
| Maddison Dunning   | Lead Developer              |
| Alec Baileys       | Developer                   |
| Benjamin Cushing   | Business Analyst            |
| Elshaday Mesfin    | Business Analyst            |
| Tyler Puschinsky   | Software Tester             |

*Table 1 - Project Stakeholders*

## 1.4 Definitions, acronyms, and abbreviations

The table below shows the acronyms, abbreviations, and definitions of the corresponding row.

| Acronyms and Abbreviations | Definitions                                    |
|----------------------------|--|
| NLU                        | Natural language understanding                 |
| STM                        | Short-term memory                              |
| IDE                        | Integrated development environment             |
| SDK                        | Software development kit                       |
| AVD                        | Android virtual device                         |
| API                        | Application programming interface              |
| GIT                        | Open-source distributed version control system |

*Table 2 - Definitions, acronyms, and abbreviations*

## 1.5 References

Table 3 below lists the references used through the development of the Harkify application and this programmer's guide.

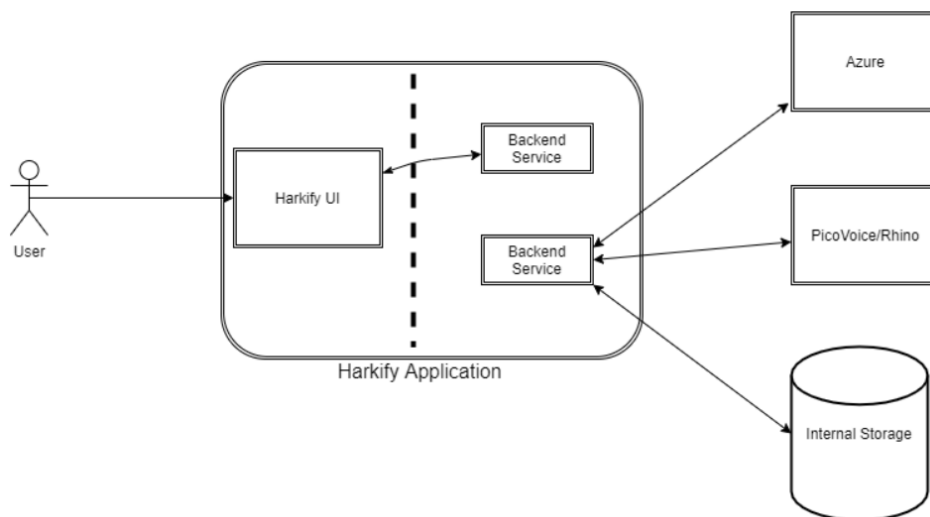
| Reference about                                      | Link  |
|--|---|
| Flutter startup and documentation                    | <a href="https://flutter.dev/docs">https://flutter.dev/docs</a>   |
| Setup and API guide for Picovoice:                   | <a href="https://picovoice.ai/docs/api/picovoice-flutter/">https://picovoice.ai/docs/api/picovoice-flutter/</a> _   |
| Documentation for porcupine                          | <a href="https://picovoice.ai/docs/api/porcupine-flutter/">https://picovoice.ai/docs/api/porcupine-flutter/</a>   |
| Documentation for Rhino                              | <a href="https://pub.dev/packages/rhino">https://pub.dev/packages/rhino</a>   |
| Development console for creating .ppn and .rhn files | <a href="https://console.picovoice.ai/ppn">https://console.picovoice.ai/ppn</a>   |
| Azure speech recognition package                     | <a href="https://pub.dev/packages/azure_speech_recognition">https://pub.dev/packages/azure_speech_recognition</a>   |
| Azure API documentation                              | <a href="https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speaker-recognition-overview">https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speaker-recognition-overview</a> |

*Table 1 – References*

## 2.0 System Architecture

### 2.1 Architectural Overview

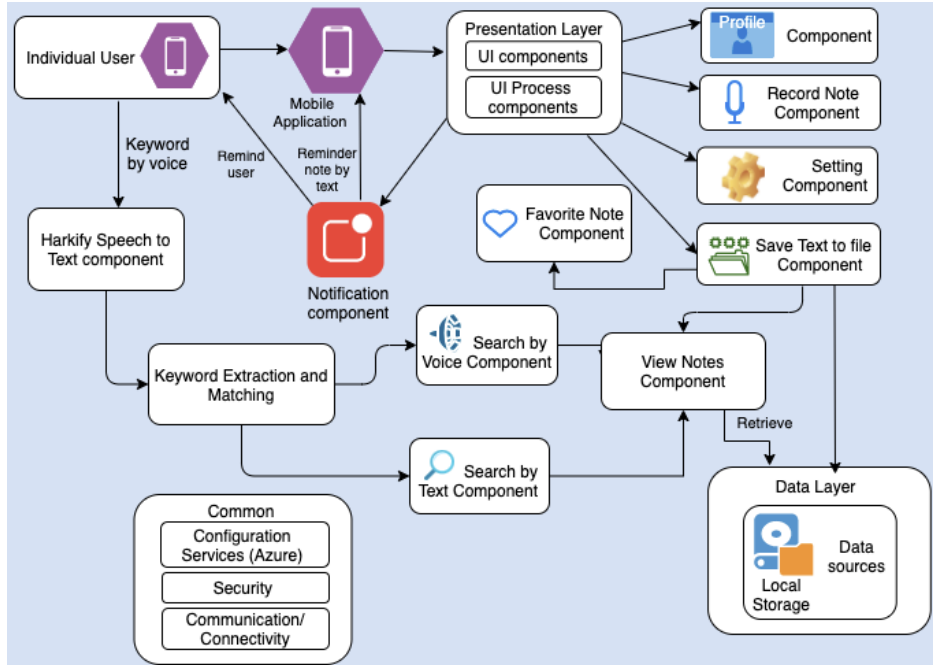
The Harkify application is comprised of 4 main components: Voice Recognition/talk to text services, Note Storage, Back-end setup services and the front-end layer. The Voice Recognition and talk to text are two separate services as both services specialize in each service. The User will interact with the UI, which will send back to any back-end service to process the request. Voice notes will be validated and analyzed before saving and storing in local storage.



*Image 1 – Architecture Overview*

## 2.2 Architectural Design

The Harkify architecture is broken down into several distinct components, as shown in the diagram below.



### Image 2 – Architectural Design

## Profile

This component is used to view & edit the user's profile details.

### Record Note

This component is where the user is allowed to record new notes by directly starting the recording from the UI.

## Settings

This component provides an interface to the Harkify application settings, where the user can configure their personal preferences.

## Save Text to file

This component provides save access to the actual data layer of the application. Both written and voice generated notes are passed through this component to be saved to local storage.



## **View Notes**

This component provides the interface for viewing the notes that are saved within the application.

## **Speech to Text**

This component handles converting speech to text within the application, the output text is sent to the Keyword Extraction and Matching component to identify user commands, as well as saving notes from converted speech.

## **Keyword Extraction and Matching**

This component infers user intents for commands to perform actions such as start recording, stop recording, search text, etc.

## **Search By Voice**

This component uses voice commands to search for notes and displays matching notes in the view notes component.

## **Search By Text**

This component provides a text search area to search for notes and displays matching notes in the view notes component.

# **3.0 Application Setup**

This section covers all software dependencies necessary before the Harkify codebase can be compiled and run locally.

## **3.1 Git and Code Deployment**

### **3.1.1 Download Git**

Git is the source control software used for development of the Harkify application. Git Bash is the command window for executing Git commands. Both are necessary for interacting with the application's codebase.

- Download URL: <https://git-scm.com/download/win>
- Download the installation package from the above URL, running it to install Git and Git Bash



Image 3 – git setup

### 3.1.2 Cloning Harkify Repository

The Harkify code repository is stored in GitHub, a Git repository website. New programmers must get access to the repository first and then use Git commands to clone it to their local machine. For a guide on how to clone the repository directly in VS Code consult section 4.4: Git and GitHub, in the Deployment and Operations Guide.

- Navigate your browser to <https://github.com/> and create a new account
- Contact the Harkify DSO representative to request access to the Harkify git repository
- Once access has been given, execute the following steps:
  - Open Git Bash
  - Navigate to the folder where you store your source control files (i.e. “c:\src”)
  - Run the following Git command to clone the full repository of source control for the Harkify application:
    - `git clone https://github.com/umgc/summer2021.bravo.git`

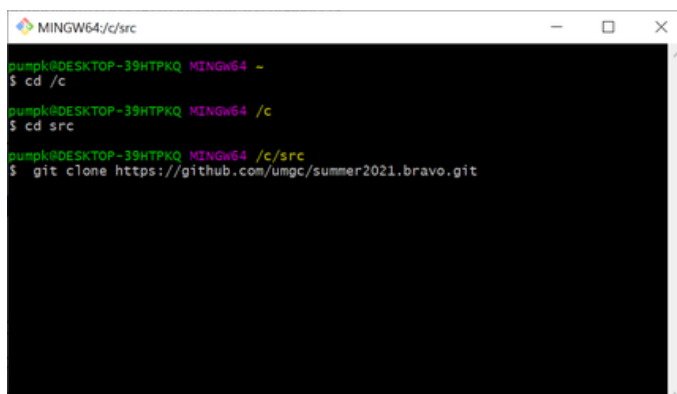


Image 4 – git terminal view

### 3.1.3 Install Flutter SDK

The Flutter Software Development Kit (SDK) allows you to develop in the Flutter language.

- Download URL: <https://flutter.dev/docs/get-started/install/windows>
- Copy the flutter folder to your machine from the downloaded .ZIP file to wherever you want to store the Flutter SDK files (i.e. "c:\flutter")
- Open your system window for "Editing the system environment variables"
- Add the "...\\flutter\\bin" folder (replace "..." with the location of your flutter folder) to the PATH environment variable for your machine so you can run flutter commands from the command line

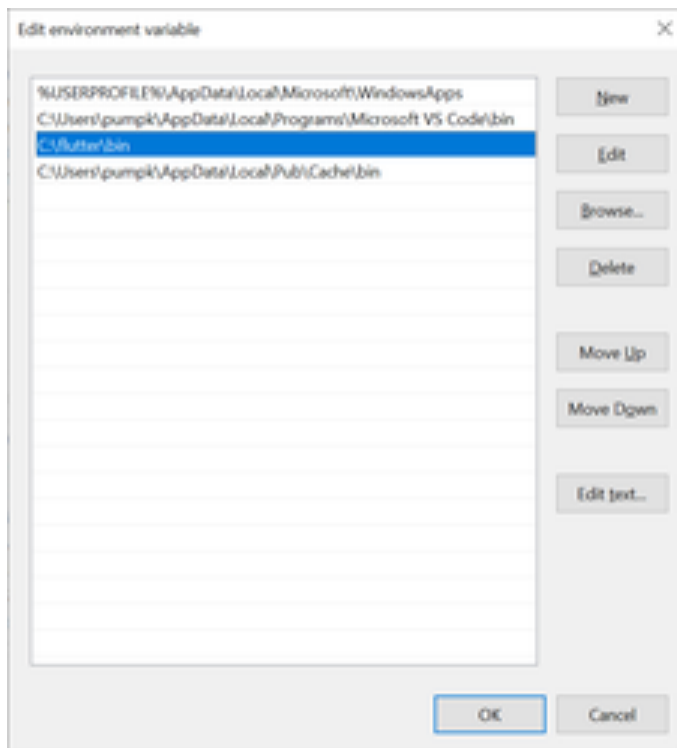


Image 5 – environment variables

### 3.1.4 Install Java 8 SDK

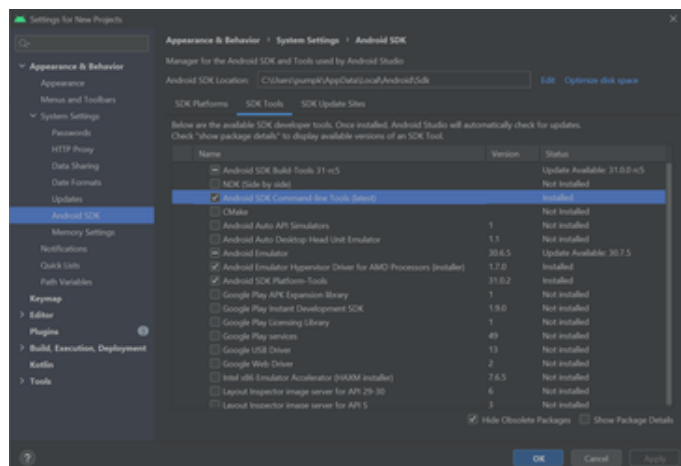
Java is required for Flutter to work correctly. Java 8 is the version of the Java SDK that works best with Flutter.

- Download URL: <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>
- If you install a later version of Java like 16, you may get this error when trying to run a Flutter app:
  - *Could not open settings generic class cache for settings file '...\\android\\settings.gradle'*

Android Studio makes it much easier to test the mobile implementation of Harkify. It includes an Android phone emulator that integrates easily with the Visual Studio Code IDE.

- 
- Welcome to Android Studio
- Android Studio
- Version 4.2.1
- + Create New Project
  - + Create New Flutter Project
  - Open an Existing Project
  - Get from Version Control
  - Profile or Debug APK
  - Import Project (Gradle, Eclipse ADT, etc.)
- Events Configure Get Help

- Once Installed, perform these steps:
  - Install the Android SDK Command-line Tools under "Configure->SDK Manager->Android SDK->SDK Tools"



8

- Install Flutter and Dart plug-ins on the same screen, using the left-nav pane’s “Plugins” link



Image 8 – Flutter and Dart install

### 3.1.6 Install Visual Studio Code

Visual Studio Code is the recommended IDE for developing the Harkify application. Please note that Visual Studio 2019 is not supported by Flutter.

- Download URL: <https://code.visualstudio.com/Download>
- Once installed, perform this step:
  - Install Flutter and Dart extensions using the “Extensions” link on the left-nav pane

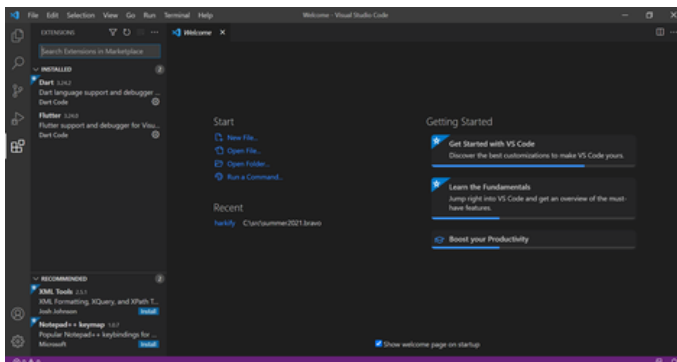


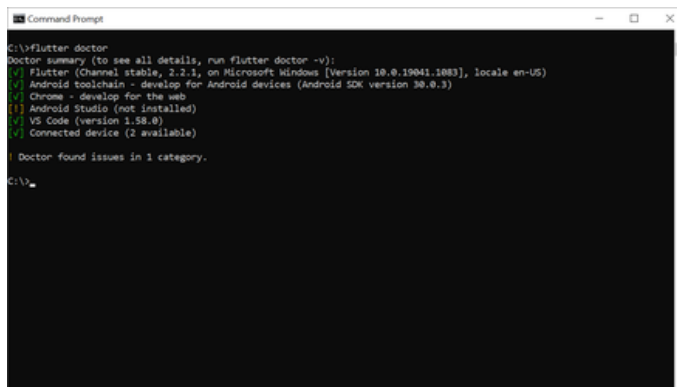
Image 9 – Visual Studio Code

### 3.1.7 Flutter Doctor Checks

Flutter Doctor is a command-line utility for checking if everything necessary is installed for Flutter to run correctly. Running it will help identify missing dependencies.

- Open a command prompt window
- Run "flutter doctor" from command-line to check if everything is installed
  - Android Studio may be listed as uninstalled even though you have installed it—you can ignore that
  - Android licenses will be listed as not having been accepted
- Run "flutter doctor --android-licenses" from command-line to accept licenses
  - Click "y" at all license prompts

Once all these installation steps have been completed, you are ready to run a Flutter application.



```
C:\>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.2.1, on Microsoft Windows [Version 10.0.19041.583], locale en-US)
[✓] Android toolchain - develop for Android devices (Android SDK version 30.0.3)
[✓] Chrome - develop for the web
[!] Android Studio (not installed)
[✓] VS Code (version 1.58.0)
[✓] Connected device (2 available)

! Doctor found issues in 1 category.

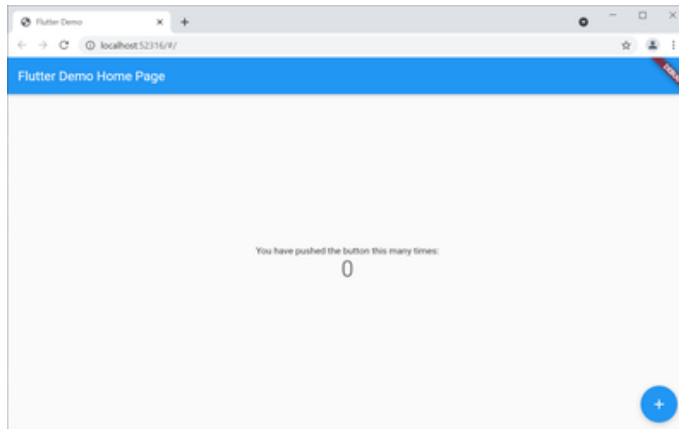
C:\>
```

*Image 10 – Terminal View: Flutter Doctor*

### 3.1.8 Test Out Dependencies

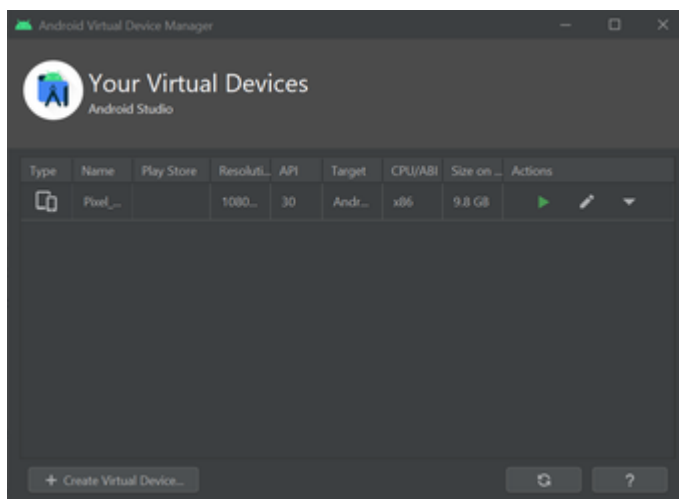
Flutter applications have many dependencies, making it worthwhile to test out the entire Flutter pipeline with the default Hello World project before attempting to run Harkify. This will help isolate installation and configuration issues from other issues stemming from the Harkify code.

- Open VS Code
- Press Control-Shift-P and type "Flutter"
- Select "Flutter: New Application Project" to create a Hello World project for testing out Flutter
- Testing in Chrome
  - Out of the box, the Flutter project will run in the Chrome browser
  - To ensure that Flutter will run on Chrome, Press Control-Shift-D and then "Run and Debug" to run the default project
  - You should see the demo home page with a round "+" button in the bottom-right corner



*Image 11 – Chrome Emulator View*

- Testing in Android emulator:
  - Android Studio provides an emulator for an Android phone to test out Flutter applications in a mobile format
  - Open Android Studio
  - Click "Configure->AVD Manager" to open the Android Virtual Device Manager
  - Press the play button on the row whose name begins with "Pixel" to start up a Pixel phone emulation



*Image 12 – Android Emulator setup*

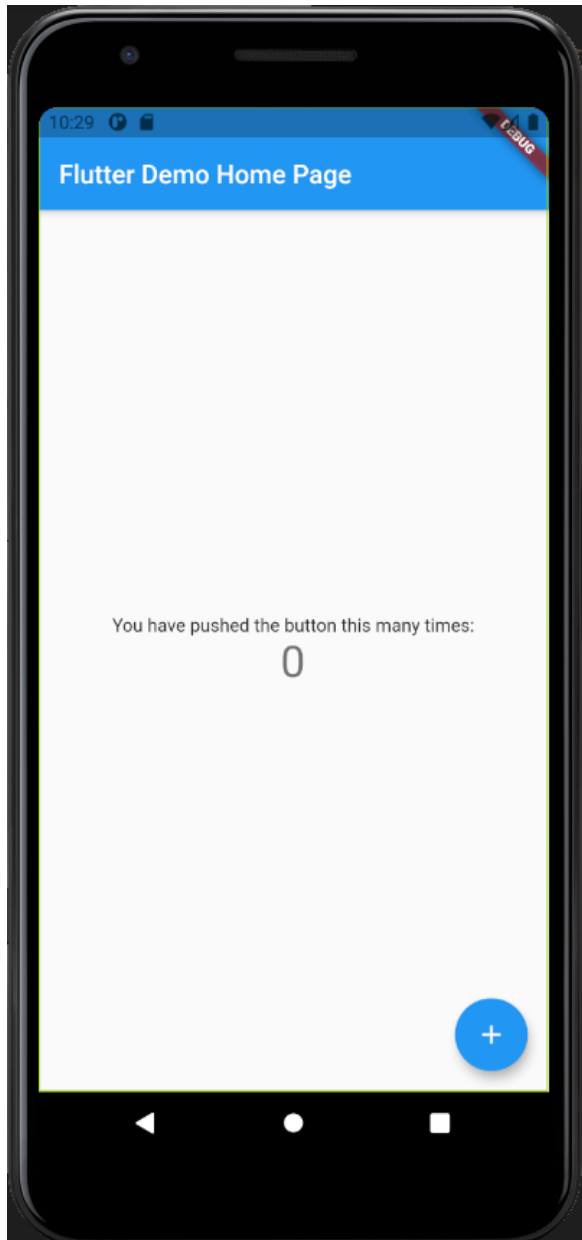
- Wait for the phone emulator to appear and boot up, showing a phone home screen



*Image 13 – Andriod Emulator View*

- If the phone screen remains black, you may need to hit the power button on the right-side menu panel next to the phone image
  - Go back to VS Code, which should automatically detect the android emulator being activated
  - You should see "Pixel\_3a\_API\_30\_x86 (android-x86 emulator)" or something similar in the blue footer to indicate your project is connected to the emulator
  - Press Control-Shift-D and then "Run and Debug" to run the default project in the emulator





*Image 14 – Application View on Emulator*

### 3.2 Picovoice Overview

Picovoice is a voice product platform for inferring user intent from user speech. Picovoice was chosen because it is more accurate than Google Assistant or Amazon's Alexa. Furthermore, it does not require internet access as the entire library can be included with the Harkify application. This avoids network latency and connectivity issues that might plague other solutions.

Once Picovoice has been trained to recognize wake words and user intents, it integrates seamlessly into Harkify. Picovoice will listen for the presence of spoken wake words. Once they are detected, Picovoice listens for user intent, comparing speech to pre-trained intent data. If the

speech is understood, user intent is inferred and Harkify is provided with the resulting intent data which it can use to perform text note operations.

More information about Picovoice can be obtained at this URL: <https://picovoice.ai/docs/>

### 3.2.1 Wake Words

Wake words are the user speech that tells Picovoice to begin operating. Once wake words are detected, subsequent speech is used for intention inferences. Picovoice uses the Porcupine engine to detect wake words. Wake words are initially trained through the Picovoice console to produce a .PPN file which Picovoice can compare with user speech. The Harkify development team has already trained for the wake words, “Okay, so...,” which is stored in the `/assets/ok_so.ppn` file in the application codebase.

### 3.2.2 Inferences

Picovoice uses the Rhino engine for inferring user intents. User speech is analyzed and compared to a pre-trained intents file. This training was done through the Picovoice console, producing a .RHN file. The Harkify development team has already trained the Rhino engine for inferences related to recording notes, searching notes, and searching user details. These inferences are stored in the `/assets/note_taker.rhn` file in the application code base.

Inferences are composed of intents and slots. Intents are the user’s commands. Intents contain a series of spoken phrases called expressions. Any user speech that matches an expression matches the overall intent. Slots, on the other hand, are the parameters for that user command. They function as variables to store the bits of user speech that define how the command should be performed.

Picovoice’s Rhino engine returns a JSON of intents and slots from analyzing the user speech. For instance, if a user wants to save a note and says, “I need to remember,” the Rhino engine should return:

```
{
  intent: "startTranscription"
}
```

This intent signifies for Harkify to begin recording the subsequent speech as a text note.

Similarly, if a user wants to search through their notes and says “What did I say about Sunday? “, the Rhino engine will return:

```
{
  intent: "searchNotes",
  slots: {
    date: "Sunday"
  }
}
```

This intent signifies for Harkify to search through all notes for any that match the date, Sunday.

Also, if a user wants to search through their user details and says, “What is my Medicare number?” the Rhino engine will return:

```
{
  intent: "searchDetails",
  slots: {
    info: "Medicare Number"
  }
}
```

This intent signifies for Harkify to search through user details for one that matches “Medicare Number.”

Many expressions are already trained for these intents, but further expressions and intents can be added through the Picovoice console. A Rhino engine cheat sheet for generating intents and slots can be found at this URL: <https://picovoice.ai/docs/tips/syntax-cheat-sheet/>

### 3.2.3 Training

If more wake word or user intent training is needed, it must be done in the Picovoice console. The console can be accessed at this URL: <https://console.picovoice.ai/>

Team credentials to access the console will be provided by the Harkify lead developer.

### 3.2.4 Integration into Harkify

The `VoiceHelper` class manages all Picovoice operations for Harkify. It is located here: `lib\voicehelper.dart`

`PicovoiceManager` is the main class in the Picovoice library. When it is instantiated, references to the pre-trained files for wake words (`ok_so.ppn`) and user intents (`note_taker.rhn`) are provided. Callback functions are also provided for when wake words and intents are detected. The `VoiceHelper` class contains a `_wakeWordCallbackDefault` method that executes when the user says “Okay, so....” The `VoiceHelper` class also contains a `_inferenceCallbackDefault` method that executes when user intents have been analyzed, providing a list of detected inferences to the method.

## 3.3 Azure Speaker Recognition Overview

Microsoft Azure provides a RESTful API for speaker voice recognition. Harkify uses this API to ensure that only the user’s speech is interpreted for commands. Before user speech is interpreted by Picovoice, the speech is sent to the Azure API for verification. Initially, a pre-recorded sample of the user’s voice is sent as a .WAV file to identify the user as the speaker to match. This is supplied as an HTTP Post command to the API’s `CreateProfile` service. Then the current

user speech is sent as an HTTP Post command to the API's VerifyProfile service. The VerifyProfile service returns similarity data from comparing the two samples.

The similarity data includes a *recognitionResult* value and a *score* value. The *score* value is a number between 0 and 1, indicating the percent of similarity. The *recognitionResult* value is either “accept” or “reject” based on whether the similarity was greater than 50%.

More details on the Microsoft Azure API for speaker recognition can be seen at this URL: <https://docs.microsoft.com/en-us/rest/api/speakerrecognition/verification/text-independent>

### 3.4 Code Structure

Flutter and Dart has been used to develop the Harkify application. Both are innovative Google technologies popular for quickly generating complex mobile applications for Android and IOS phones. Flutter and Dart can also be used to generate web and desktop applications from the same codebase. This will extend Harkify's reach to as many platforms as possible.

#### 3.4.1 Source Code

The Harkify codebase is mostly composed of widget files and service files. The widget files represent pages that users can visit. The service files represent common operations used by more than one page. Both are saved in the `\lib` folder. Unit test files for all widgets are services are saved in the `\test` folder. External package dependencies are stored in this file: `\pubspec.yaml`

The following widgets are included:

- **Main:** Home page for the application with all navigation options
- **RecordNote:** Allows user to record their speech as a written text note
- **SaveNote:** Allows user to save a written text note
- **ViewNotes:** Allows user to view and search from all saved text notes
- **NoteDetails:** Allows user to view, change or delete a specific text note
- **ViewSettings:** Allows user to view and change application settings
- **RecordDetails:** Allows user to record their speech as a user detail
- **UserDetails:** Allows user to view, change or delete all user details

The following services are included:

- **TextNoteService:** Manages all file I/O operations for text notes and user details
- **UserSettingService:** Manages all file I/O operations for user settings
- **VoiceHelper:** Manages all Picovoice operations
- **AzureHelper:** Manages all Azure speaker recognition API calls
- **BaseMenuDrawer:** Provides a generic menu widget to be used by all pages

### 3.5 Data Security

Due to the sensitive nature of the user data stored by Harkify, which may be classified as Personally Identifiable Information (PII), data security is of paramount concern. Consequently, all data stored on the user's phone is encrypted, including text notes and user settings.

The flutter `encrypt` package is used for all encryption services. AES 128-bit encryption is the industry standard and has been employed in Harkify. A random, 128-bit encryption key is hard-coded in the `TextNoteService` class (located at: `lib\textnoteservice.dart`). Although this means that all instances of Harkify will use the same encryption key, the key will be inaccessible to users since it is compiled into the application. The disadvantage of this approach is that text notes will not be directly accessible to user outside of the application, but the increased security outweighs this disadvantage.

## 4.0 Running the Application

### 4.1 Installing Git

Git is the source control software used for development of the Harkify application. Git Bash is the command window for executing Git commands. Both are necessary for interacting with the application's codebase.

- Download URL: <https://git-scm.com/download/win>
- Download the installation package from the above URL, running it to install Git and Git Bash

The repository can be cloned by retrieving the code from GitHub. Branches can be made to make code changes and then submitted through the pull request.

### 4.2 Installing Flutter and VS Code

Harkify is written using flutter, and collaborators and developers use VS Code as the main ide to compile code. To get the latest version of Flutter navigate to <https://flutter.dev/docs/get-started/install/windows> and choose the option to download the latest Flutter SDK. Please take note of the system requirements for Flutter from the Configurations section. Install Microsoft VS Code from <https://code.visualstudio.com/Download> and choose the proper download for your system.

Create a folder on C: to store the Flutter directory. After unzipping Flutter to this directory, copy the file path and create a path to this directory in Environment Variables. For a guide on how to do this, consult section 4.1.1: Installing Flutter and VS Code, in the Operations and Deployment Guide.

After installing and starting VS Code, make sure to install the Flutter plugin. This will also install the Dart plugin, both of which are necessary to create, and run Flutter projects using the Dart language.

### 4.3 Installing Android SDK

Android Studio is required for the SDK that allows VS Code's Android emulator to function. Navigate to <https://developer.android.com/studio> and choose Download Android Studio. Follow the installer's default recommendations, especially, the installation location. Run the command "flutter doctor," ensuring first that Flutter has already been installed. With Flutter Doctor, the programmer can check to see if all requirements are met for developing in Flutter. If the Android toolchain does not have a check mark, creating a path to the Android SDK may be necessary.

First find the local path to the Android SDK, by default it should look like this C:\Users\\*yourprofilename\*\AppData\Local\Android\Sdk\cmdline-tools. Copy the path and edit the "path" user variable, create a new entry, and paste the Android SDK path. Make sure the path includes "cmdline-tools" or errors may occur. For more information about setting the path for the Android SDK consult section 4.1.3: Verify with Flutter Doctor, in the Deployment and Operations Guide.

### 4.4 Voice Recognition Services

Picovoice is a library that can be added through pubspec.yaml as a dependency. Once into the application, a pico-manager is created to assist in development. Azure Speech Recognition is accessed using REST API to create a profile of the user's voice. No packages need to be imported for the Harkify application. A valid Azure account must be used and a license for the application is provided and not expired.

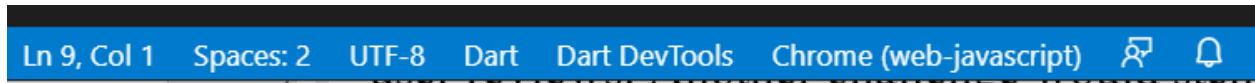
### 4.5 Harkify Application

By following the procedure, Harkify can be deployed locally and tested locally through pulling from the repository. To be able to achieve higher environments, the code is sent to GitHub and sent to DevSecOps team for further deployment. The Harkify project repo is hosted on GitHub at <https://github.com/umgc/summer2021.bravo>.

To clone the repository, select the Source Control menu on the left-hand bar in VS Code. Select Clone repository and enter the repository URL stated above. Choose a directory for the cloned repository and select "Get packages" if prompted. It may be necessary to run the "dart pub get" command in the terminal. For more information on cloning the repository, consult Deployment and Operations Guide section 4.4.2.

To run Harkify in VS Code an Android emulator is required. To select the Android emulator check in the bottom-right corner to select a device to deploy to. By default, this may have

Chrome selected.



### *Image 15- Chrome Deployment*

Click on whatever is present in this box, and select an Android device, if none is present, please refer to section 4.3 in this document or sections 4.1.2 and 4.1.3 in the Deployment and Operations guide.

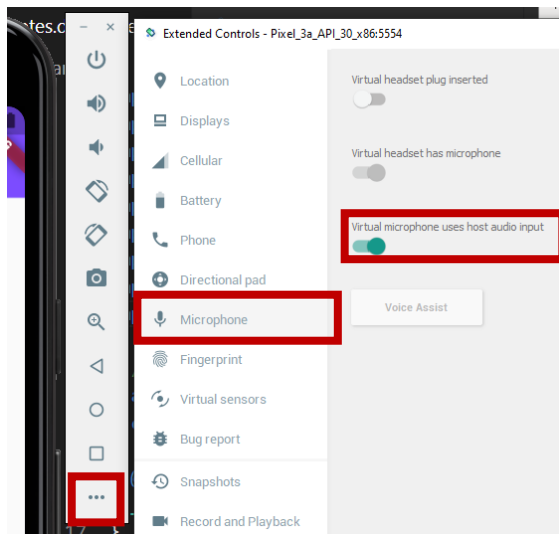
When the emulator is properly selected, the right of the bottom bar should look like this.



### *Image 16 – Android Deployment*

Now you may run Harkify on the emulator by going to Run > “Start Debugging” or “Run Without Debugging.”

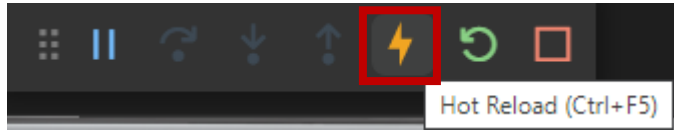
Harkify needs voice input to function, and the emulator must be configured. On the right-hand side of the emulator there is a vertical list of icons, select the three dots on the bottom. In the menu that appears, select Microphone, and ensure that the “Virtual microphone uses host audio input” setting is enabled as shown below.



### *Image 17 – Android Emulator Settings*

When Harkify runs for the first time, the user will be prompted to allow or deny microphone permissions, select allow to begin using Harkify to record notes.

Changes can be made to the code and can be applied without having to rebuild the application. With the application running, pressing the Hot Reload button, followed by the restart button, to the right will reload the application with the new code, without having to wait for it to build again, this can save a lot of time and is perfect for making small changes to the UI.



*Image 18 – Flutter Hot Reload*

## 5.0 Challenges/Issues/Concerns

Harkify is an application dedicated to help users with memory deficiencies. The interaction of users with memory loss and applications has not been fully studied and comprehended. So, our team will need to dedicate time to be able to improve usability and functionality for those that need this application the most. When the application becomes available, it could change due to user reviews. Another challenge would stem from requiring the user to be online to be able to use the recording functionalities. While Azure is the top-of-the-line system to identify voice, it requires an API connection. Slowly, there has become less issues with this requirement as many phones come with data plans that cover a large area of places. This issue however can be eliminated by using a voice identification system that is a library or internal to the device. This would eliminate the problem entirely and one that the team is considering and looking into.

Challenges in development, come in the form of testing the IOS compatibility system. Everything is developed using the android studio simulator to be able to see if code is compiling correctly. Some ways around this issue can be to use a VM or a Mac for the IOS to allow the system to test out the code using VS Code compilation system. Using a spare iPhone for development may also be used for developing Harkify on iOS.

## 6.0 Additional Help

If any assistance is required, please contact the UMGC Graduate Software Engineering department. The DevSecOps team will be able to help with trouble connecting to GitHub and other team members can help with setup. Any other information can be provided by the team members listed on this document.



## 7.0 License Information

Visual studios code is a free IDE provided by Microsoft. Any license if provided under the MIT licensing agreement at <https://github.com/Microsoft/vscode>.

Since Azure is an outside service connected through an API, the licensing must be approved. For more information on their rules and regulation for Azure, the licensing agreement can be found at <https://azure.microsoft.com/en-us/support/legal/>.

All other software used has been open source to reduce the cost of the developed application.