

GNG 2101[C]
Design Project User and Product Manual

Image Descriptor

Submitted by:

4TexttoSpeech, C2.3

Shahid Awati, 300015213

Serigne Saliou Mbacke Sourang, 300040876

Jathushan Karthigesar, 300060617

Valentin Mugabo, 300038960

April 10th, 2022

University of Ottawa

Table of Contents

Table of Contents	ii
List of Figures	iv
List of Tables	vi
List of Acronyms and Glossary	vii
1 Introduction.....	1
2 Overview.....	2
2.1 Conventions.....	3
2.2 Cautions & Warnings	3
3 Getting started.....	4
3.1 Configuration Considerations	7
3.2 User Access Considerations	8
3.3 Accessing/setting-up the System.....	8
3.4 System Organization & Navigation	9
3.4.1 The Splash Screen.....	9
3.4.2 The Onboarding Screens.....	9
3.4.3 The Main Screen UI.....	9
3.4.4 The Text Detail and Speech UI.....	9
3.5 Exiting the System	9
4 Using the System	10
4.1 Main UI.....	10
4.1.1 Image picker option	10

4.1.2	Camera access option.....	10
4.1.3	File picker option	10
4.2	Text Detail and Speech UI	10
4.2.1	Image rendering	11
4.2.2	Text rendering	11
4.2.3	Voice output.....	12
4.3	Splash screen and Onboarding screens	12
5	Troubleshooting & Support	13
5.1	Error Messages or Behaviors	13
5.2	Special Considerations	15
5.3	Maintenance	15
5.4	Support	16
6	Product Documentation	17
6.1	Final Prototype	18
6.1.1	BOM (Bill of Materials)	18
6.1.2	Equipment list	18
6.1.3	Instructions.....	18
6.2	Testing & Validation.....	25
7	Conclusions and Recommendations for Future Work	27
8	Bibliography	28
	APPENDICES	30
9	APPENDIX I: Design Files	30
10	APPENDIX II: Other Appendices	31

List of Figures

Figure 1: Prototype overview.....	2
Figure 2: Splash/Welcome Screen	4
Figure 3: Onboarding Screens	5
Figure 4: Main UI	6
Figure 5: Gallery access, Validate or Retake a taken picture, File access.....	6
Figure 6: Image and Text rendering to Text Detail and Speech UI.....	7
Figure 7: Downloading the apk file and viewing the app	8
Figure 8: Text rendering to Text Detail and Speech UI for Large Texts.....	11
Figure 9: iOS testing error 1	13
Figure 10: iOS testing error 2	14
Figure 11: iOS testing error 3	14
Figure 12: DOCX file selection exception error.....	15
Figure 13: Operating System selection for Flutter installation	19
Figure 14: First step in the Flutter installation on Windows	19
Figure 15: After installing flutter.....	19
Figure 16: Android Studio download	20
Figure 17: Running a command in the command prompt	21
Figure 18: flutter extension installation on VS code	22
Figure 19: Starting the emulator	22
Figure 20: Pixel 3 emulator.....	23
Figure 21: Project tree lib folder part 1	24

Figure 22: Project tree lib folder part 2	24
Figure 23: Project tree lib folder part 3	24

List of Tables

Table 1. Acronyms.....	vii
Table 2. Glossary	viii
Table 3: Bill of Materials	18
Table 4. Referenced Documents	30

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
UPM	User product manual
UI	User interface
MB	Megabytes
apk	Android application Package
PNG	Portable Graphics Format
JPEG	Joint Photographic Experts Group
TXT	Text file
OOP	Object orientated programming

Table 2. Glossary

Term	Acronym	Definition
User product manual	UPM	User product manual that provides the instructions of use of 4Text2Speech.
User interface	UI	User interface is the series of screens, pages, and visual elements.
Megabyte	MB	A unit of information.
Android application PacKage	apk	An application file ready for installation on an Android device.
Joint Photographic Experts Group	JPEG	The type of file supported by 4Text2Speech.
Portable Graphics format	PNG	The type of file supported by 4Text2Speech.
Text file	TXT	The type of file supported by 4Text2Speech.
Object oriented programming	OOP	A computer programming model that organizes software design around data, or objects, instead of functions and logic.

1 Introduction

This User and Product Manual (UPM) provides the information necessary for users who experience partial visual impairment to effectively use 4Text2Speech and for prototype documentation. The context of this project is based on a client who has visual impairment and who is not satisfied with the available solutions on the market to help them read text out loud. This has led to the development of a mobile application unlike any other in the market, that would meet the client's most critical requirements. The mobile application will not collect any personal data from the user as input to avoid any conflicts of interests. This document is organized in such a way to make it easy for anyone reading it to understand. It starts off by providing a general overview of the final prototype before giving the user a deeper view of the mobile application. It then explains to the user how to use the system and provides them with any troubleshooting that they may go through. Furthermore, it provides a detailed documentation of the product itself and the way it was built. Finally, some conclusions and recommendations for future work are brought to the attention of the reader.

2 Overview

The problem revolved around coming up with a user-friendly mobile application that could help our client who currently faces symptoms of visual impairment and has a difficult time reading pieces of text on their mobile device. Therefore, solving this issue with care is very important since there is a large population of the world who undergo partial visual impairment and there aren't many effective solutions out there in the market.

The fundamental needs of the user included the mobile application to have the ability to read out loud text from an image that would be easy to use and hold minimal costs. Additionally, the mobile application should be able to allow the user to select common file types from their device and have access to their camera to take pictures of text at a convenient reading speed for the reader. Lastly, the remaining needs of the user were to develop the mobile application

When comparing 4Text2Speech with other similar products in the market, our product excels in providing the users with a user-friendly environment that allows the user to transcribe the texts from images, various file types and images taken from their mobile camera at free of cost. Whereas other products in the market are not as user friendly and often charge users a subscription fee to use some features in the mobile application.

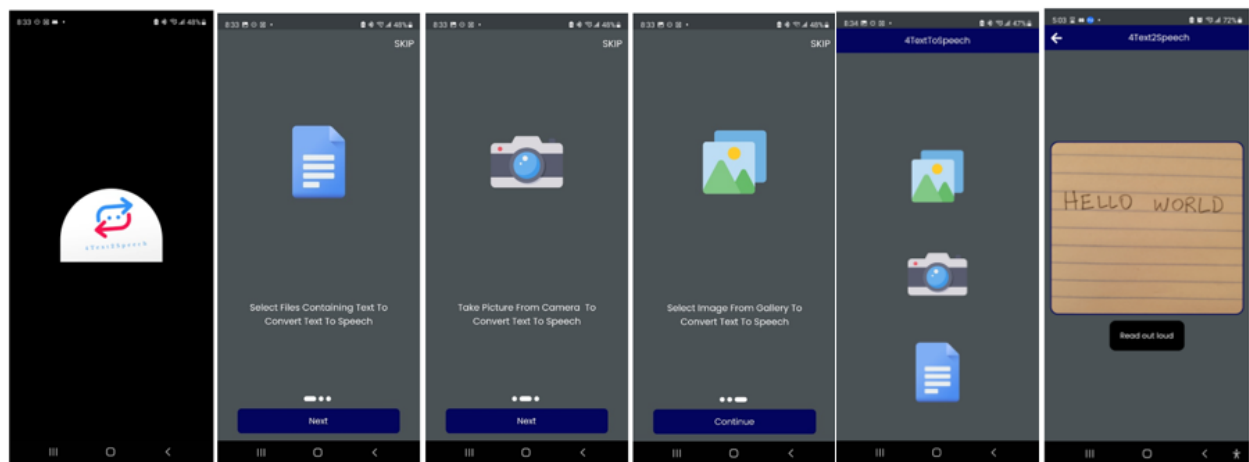


Figure 1: Prototype overview

2.1 Conventions

This is not applicable.

2.2 Cautions & Warnings

The user must have an android device so that they can download the apk file of the 4TexttoSpeech application and 41 megabytes of storage.

3 Getting started

The first screen the user is presented to when they open the application is the splash screen. The latter is a welcome screen containing the logo of the company and is intended to welcome the user. Furthermore, provided the user has downloaded and is opening the application the first time. They are presented with the onboarding screens. The latter are meant to serve as a tutorial for the user. There are 3 different onboarding screens, each one showing an application feature icon and its intended function. A button at the bottom of each of those screens, except the last one, shows “Next” and allows the user to move on to the next onboarding screen. “Continue” instead of “Next” at the bottom is shown on the last onboarding screen for the transition to the main or home User Interface (UI). Alternatively, another button is shown at the top right, “SKIP” if the user does not want to go through the tutorial. The following images illustrate what has been described above:



Figure 2: Splash/Welcome Screen

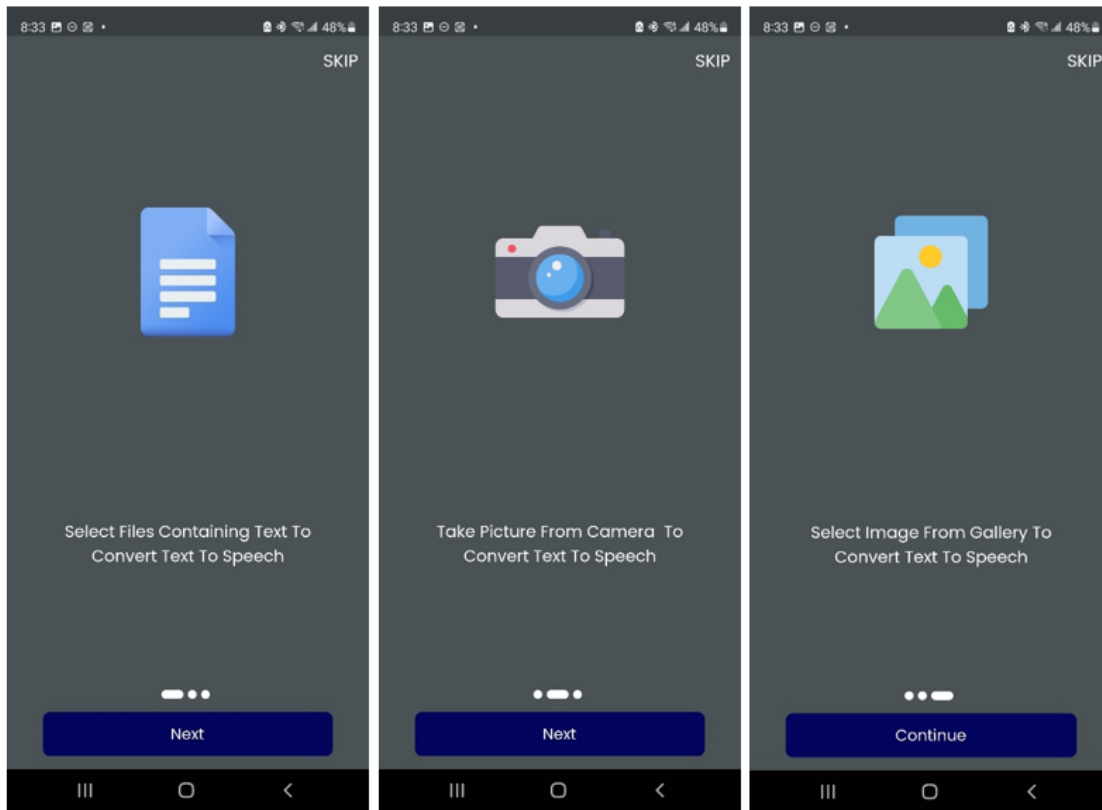


Figure 3: Onboarding Screens

After the onboarding screens, the user is then presented to the main or home UI. In the main UI, the three different options the user can choose from are shown, each one performing the function as described on the onboarding screens. The first option is a gallery icon, which when the user taps on, they have access to their gallery photos on their device and they can select the image they are interested in reading the extracted text out loud. The second option is a camera like icon which when the user taps, they can take a picture of some text they want the application to read out loud. The supported images are of type PNG and JPEG. The last option is a file icon which when the user taps, they can select between the files on their device. Again, the supported files are PDF and TXT. Depending on whether the extracted text that is read out loud is from an image or from a file, the behavior is different. If an image is chosen from the device or a picture is taken using the camera, the image is rendered on the Text Detail and Speech UI. In the case of taking a picture, the user has the option of retaking it if it was not clear enough for example. They can do that by tapping on the “Retry” button or they can proceed by tapping on the “OK” button. If a file is chosen, as the image could not be rendered in the case of multiple pages, instead, the text is extracted and output on a small rectangle box in the Text Detail and Speech UI. The button named “Read out loud” must be pressed by the user to start reading the text out loud. Below are images which illustrate what was described above:



Figure 4: Main UI

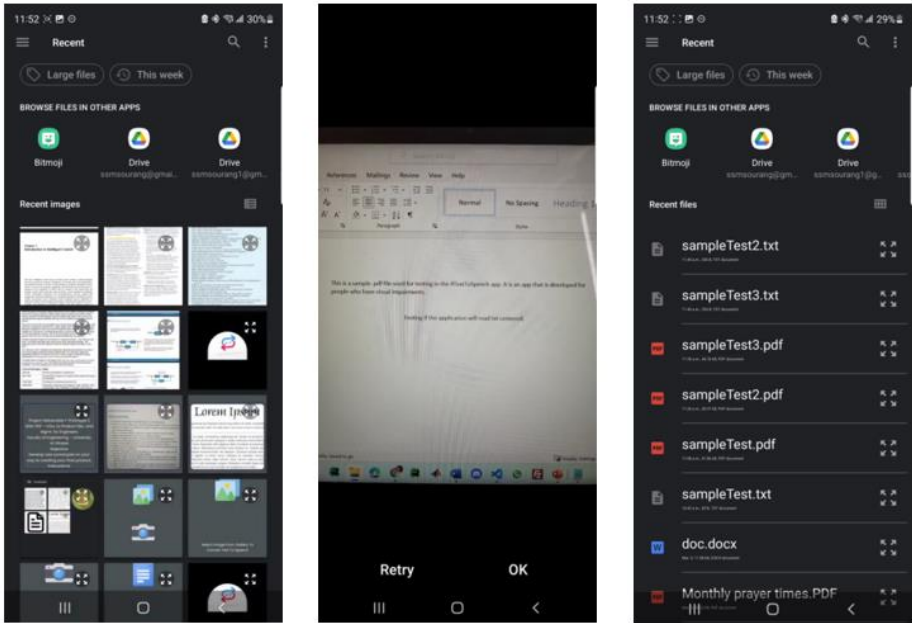


Figure 5: Gallery access, Validate or Retake a taken picture, File access

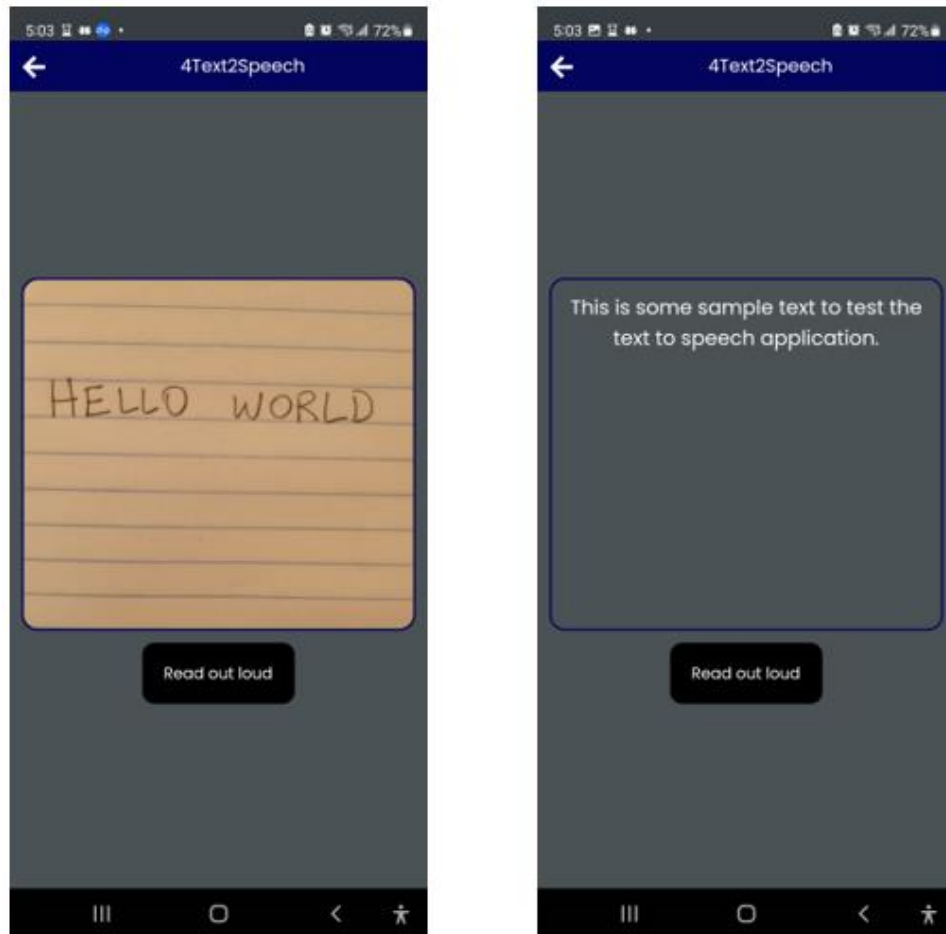


Figure 6: Image and Text rendering to Text Detail and Speech UI

There are no additional or special steps the client must take in order to get the prototype to function properly. The mobile application was designed to be as simple to use as possible.

3.1 Configuration Considerations

The application runs on Android devices such as smartphones or tablets. The device needs to have a great quality speaker built-in for good sound quality output. The corresponding apk file of the application can be downloaded at this link: [app](#). The application does not require a lot of memory space and takes about 41 MB. Upon downloading the file, the application will then appear in the device wherever you usually access your other apps. It will usually be shown among your first apps provided the device displays apps in an alphabetical order. Below is an illustration to downloading the apk file in the device and the app name and icon is also highlighted in blue:

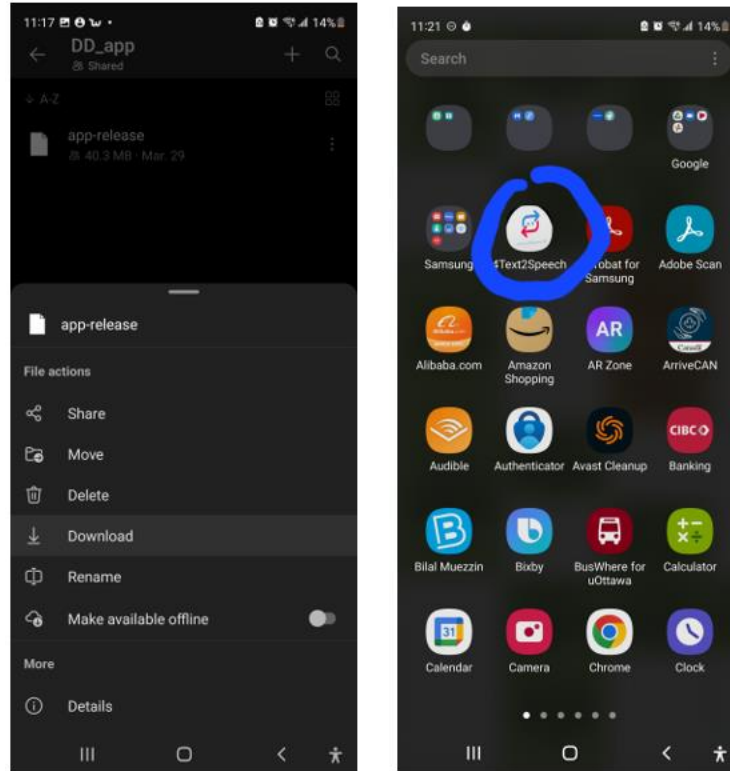


Figure 7: Downloading the apk file and viewing the app

3.2 User Access Considerations

Any android user has the ability to access the application and use it. iOS compatibility is still in development and testing and further notice will be provided when it becomes available. An internet connection is not required to use the application and therefore the user can convert text to speech wherever they are.

3.3 Accessing/setting-up the System

The user has one of two choices for accessing and setting up the system. They can either download the apk file at this link: [app](#) or they can reach out to one of the developers: ssour081@uottawa.ca, sawat018@uottawa.ca, jkart022@uottawa.ca and vmuga065@uottawa.ca who will send them the apk file to be downloaded on their android device.

3.4 System Organization & Navigation

The application is simply organized, and the navigation could not have been easier.

3.4.1 The Splash Screen

This is a welcome screen the user is presented to every time they enter the application. It shows the application logo on a black background.

3.4.2 The Onboarding Screens

This is a series of screens (3 in total) which only show the very first time the user enters the application. It is meant to serve as a tutorial for them to know the function of their different options.

3.4.3 The Main Screen UI

It consists of the screen in which the user has one of three choices: selecting an image (PNG, JPEG) from their device, accessing their camera to take a picture, accessing a file from their device (PDF, TXT).

3.4.4 The Text Detail and Speech UI

This is where the text is extracted and rendered in a rectangle box if it is file. If it is an image, the image is shown on the rectangle box as well. Furthermore, the user can tap on “Read out loud” to start converting text to speech and they can tap “Stop” when they want to stop the reading.

3.5 Exiting the System

The user can turn off or exit the application by tapping on to the home button of their device. If the user wants to completely close it, they can access their recent apps by tapping on the home menu on the bottom left of their device, and by swiping up the “4TexttoSpeech” application like they would for any other app.

4 Using the System

The system was designed to be very simple and easy to navigate. This way, our mobile application will have a competitive advantage over its competitors. Our client who was experienced with similar apps gave us a high rating and was very satisfied with the over system. It is minimalistic and has an elegant, simple and effective design that surpasses many of its competitors. This design makes 4Text2Speech very reliable. The following sub-sections give detailed, step-by-step instructions on how to use the various functions or features of the 4Test2Speech mobile application.

4.1 Main UI

The functions of the mobile application are presented in the main UI shown in Figure 3. In the main UI, you have the option to each one performing the function as described on the onboarding screens.

4.1.1 Image picker option

The first option is the image picker option. It is represented by a gallery icon. From this option, the user accesses their gallery photos on their device to select the intended image for reading. The user must select a JPEG file or a PNG file.

4.1.2 Camera access option

The second option is the camera access option. It is represented by a camera icon. From this option, the user takes a picture of any text for reading. The last option is a file icon where the user can choose text files on their device.

4.1.3 File picker option

The last option is the File picker option. It is represented by a file icon. From this option, the user can choose text files on their device. The user must select a TXT file or a PDF file.

4.2 Text Detail and Speech UI

After the user performs one of the options mentioned in section 4.1, the application will render the chosen/taken picture or render the extracted text from a supported file. This brings us to the interface demonstrated in Figure 7, which is the “Text Detail and Speech UI”. As shown in Figure 7, there is an arrow in the top left, where the user can return to the previous interface by clicking on it.

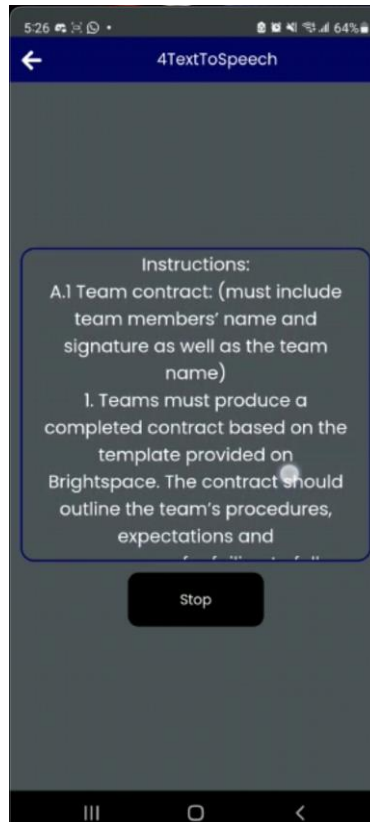


Figure 8: Text rendering to Text Detail and Speech UI for Large Texts

4.2.1 Image rendering

First, the user has the option to retake a picture if it wasn't clear enough. To retake a picture, the user must press the “Retry” button or he/she may continue by pressing the “OK” button. Moreover, if the user chooses an image type other than JPEG and PNG, an error message is displayed.

4.2.2 Text rendering

Second, in the case where the user chooses a file with multiple pages, the text is extracted and output on a rectangle box in the Speech UI. Moreover, if the user chooses a file type other than PDF and TXT, an error message is displayed. This error message is “file not supported”. Then, the user can choose another file type or close the app.

4.2.3 Voice output

Once the user chooses and performs one of the options mentioned in section 4.1, the application will render the chosen/taken picture or chosen text. Then, the user will have to press the “Read out loud” button for the app to read the text out loud. This step is demonstrated in Figure 7. Once you stop the reading by clicking on the Stop button and play again, the app will read the extracted text from the beginning.

4.3 Splash screen and Onboarding screens

These functions were detailed in Section 3.

5 Troubleshooting & Support

There were various challenges during the app development but most managed to be resolved except for the app deployment on the iOS platform and the unsupported docx file type.

5.1 Error Messages or Behaviors

As flutter allows the application to be downloaded on both iOS and Android, it needed to be tested on both platforms during the app testing stage in both 2nd and final prototype, but due to (internal errors), the testing on iOS were not successful.

When the code is ran or executed on an iOS device such as a MAC, 2 error messages appear in red. These are:

- Error running pod install
- Error launching application on device (iPhone SE in this case)

When trying to debug the errors, some answers on stackoverflow, a platform for sharing programming issues and solutions on how to solve them, seemed to all converge to a common cause for the problem. That is, most answers seemed to suggest that versions of flutter and cocoapods, the latter being specific to iOS, need to be up to date. However, when attempting the suggested steps to resolve the issue, they were not successful. After some more digging, it was found that another factor is that apple has more security requirements as compared to Android. That is, they need the user to sign in with their apple ID meaning that the developer requires to have a sign in certificate to run a code on iOS emulator.

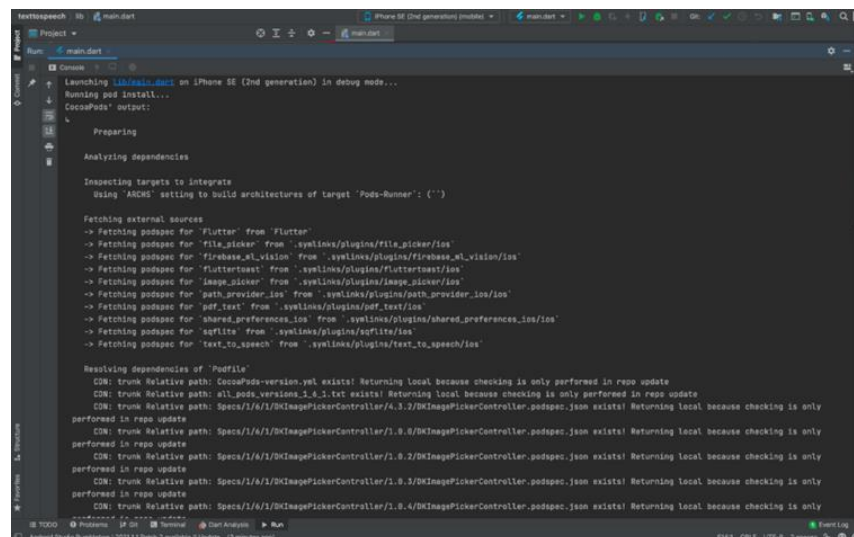


Figure 9: iOS testing error 1

5.2 Special Considerations

The app doesn't support a docx file type and when chosen, it displays a "File isn't select" error prompting the user to choose another file type.

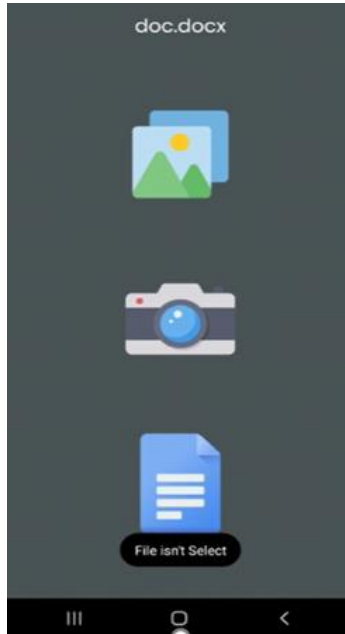


Figure 12: DOCX file selection exception error

Furthermore, the app also will not always read complex pictures with text and randomly distributed illustrations/images in them.

5.3 Maintenance

To avoid app failure, regular maintenance should be performed in form of consistently testing the application to make sure that all features are fully functional and that all potential exceptions have been caught and managed. Apart from these regular tests, the application is also going to be updated, by bringing in new features that were not implemented due to a lack of time. Those features will be added one at a time with an appropriate amount of testing to limit errors while still focusing on a high level of user experience satisfaction.

5.4 Support

Users can get emergency assistance and system support by directly sending an email to one of the developers. Their email contacts have been provided as part of section 3. Ideally, a chatbot for directly helping clients is preferred but since the website for our app is still under construction, identified problems can be reported by sending out an email for support.

6 Product Documentation

The final prototype was built using flutter, an open-source software development kit and the code base is in the dart language. Many options were available to build the final prototype. For instance, MIT App Inventor was one of the first choices as it is a blocks language which would make it very beginner friendly to build the application. However, the use of APIs which would allow to convert text to speech for instance would be difficult to integrate and there was hardly any documentation regarding that matter. Another design consideration was Kivy, a python framework to build cross-platform mobile applications. The issue with it is the lack of resources to help get started as Python's force is not mobile development, although being a general-purpose programming language.

Flutter is known for mobile development, which means, there is much more documentation and resources, making it beginner friendly and more feasible than other options. Furthermore, a critical assumption was that setting up flutter would not be much different on PC from MAC. However, setting up on MAC was harder as some sign-in certificate was required, which took a lot of time trying to figure out the issue and slowed the development for almost 10 days. The client was contacted to let them know the issue and they said to shift the focus to Android instead of iOS, as the client had an iOS device but would also be getting an Android device. The time that was lost meant more design considerations and adjustments needed to be made. That is, due to a lack of time, the focus was now put on the most critical and prioritized user needs, and the rest of the features would be added depending on the remaining time. All the design considerations when building the application were from the client's feedback. Namely, the high contrast of the application, the fact that it should be very simple to use or the fact that it should have a convenient speech reading speed.

When building the application initially, it was done in a functional programming way. This means that all the functionality was being contained into one main design file. However, it was quickly clear that as more functionality was being added, the file was becoming bigger and bigger, and it would be hard to navigate between the different functionalities using this approach. Another issue is that this approach led to more bugs as the different components were not separated. After reading through the documentation and looking at the best practices when it comes to building a large or medium scale applications like these one, another approach was used. The different components and features of the applications were now separated into their own files, making it much easier to navigate. The fact that coding the application was approached using object-oriented programming (OOP) was beneficial because the code became much more organized, and it was less error-prone.

6.1 Final Prototype

6.1.1 BOM (Bill of Materials)

Table 3: Bill of Materials

Part Name	Description	Quantity	Cost (\$)	Link
Flutter	An open-source UI software development kit that allows developers to develop applications for Android, iOS and many more platforms.	1	0	Install Flutter
image_picker	Allows to select different images from the gallery.	1	0	image_picker Flutter Package (pub.dev)
file_picker	Allows to select between files from the device's filesystem.	1	0	file_picker Flutter Package (pub.dev)
text_to_speech	Extracts the text from an image and reads it out loud.	1	0	flutter_tts Flutter Package (pub.dev)

6.1.2 Equipment list

- Laptop (at least 8 GB of RAM)
- MAC (at least 8 GB of RAM)
- Android Studio ([Download Android Studio and SDK tools | Android Developers](#))
- Git ([Git - Downloads \(git-scm.com\)](#))
- Visual Studio Code ([Download Visual Studio Code - Mac, Linux, Windows](#))

6.1.3 Instructions

There are several steps to follow in order to reproduce building the mobile application. The first step is to install flutter depending on one's operating system by accessing this link: [Install | Flutter](#). The user then needs to choose between the different operating systems as shown in the picture below:

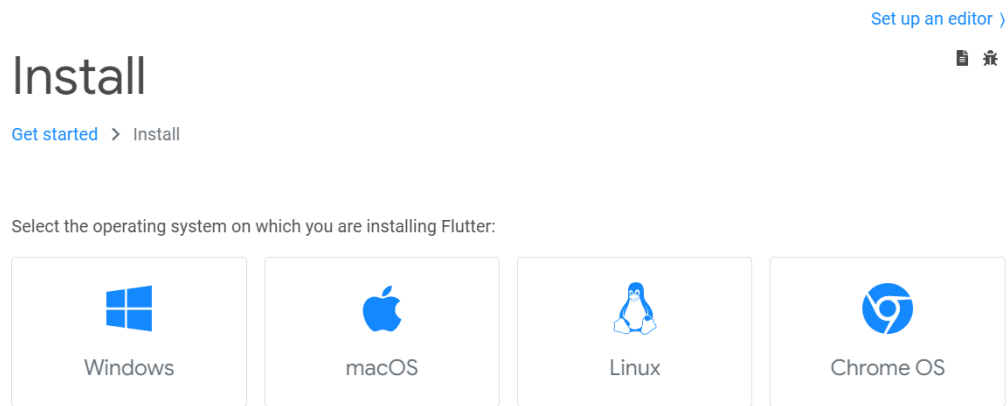


Figure 13: Operating System selection for Flutter installation

Once the user clicks on their operating system, they are then directed to the appropriate installation steps. For example, selecting Windows will get them to a page that says “Windows Install” at the top and further down, “Get the Flutter SDK”:

Get the Flutter SDK

1. Download the following installation bundle to get the latest stable release of the Flutter SDK:

`flutter_windows_2.10.4-stable.zip`

For other release channels, and older builds, see the [SDK releases](#) page.

Figure 14: First step in the Flutter installation on Windows

The user must then follow all the steps listed on the web page until they see the following at the bottom of the page:

Next step

Set up your preferred editor.

[Set up an editor >](#)

Figure 15: After installing flutter

“Select an editor” as shown on Figure 9 should then be clicked to install an editor. A lot of editors are available to develop a Flutter application, but the most common ones are Visual Studio Code and Android Studio. On the “Select an editor” page, a link to download Android Studio will be provided: <https://developer.android.com/studio>. Clicking on download options will bring you down here:

Android Studio downloads

Platform	Android Studio package	Size	SHA-256 checksum
Windows (64-bit)	android-studio-2021.1.1.23-windows.exe Recommended	872 MiB	def7755842942be93c56db94c7d53eed70004b0ab1dc8883e959da0649032582
	android-studio-2021.1.1.23-windows.zip No .exe installer	882 MiB	a4d9bf8f4f67392052d70770495e572eda4d85d7a3d59c648e655a1b9d2aea27
Mac (64-bit)	android-studio-2021.1.1.23-mac.dmg	929 MiB	ce85ddff4c39f3eaa0b314cffa5f23987e255cce5d7aa900281884643f0a2db3
Mac (64-bit, ARM)	android-studio-2021.1.1.23-mac_arm.dmg	926 MiB	f2ab09466927f338b2c1ff27635be7cf24b08191bccdeb63ee68b33a01d0a05c
Linux (64-bit)	android-studio-2021.1.1.23-linux.tar.gz	904 MiB	b37506cd8ac7a80fe30cd1724e3be5c2d970a7aa6aa3fc9ca745afe3700aabc
Chrome OS	android-studio-2021.1.1.23-cros.deb	761 MiB	f5733f44ab39d61b8885abc0d49b69a191d505114c154f3b5c4ee4d2f0b233e6

Figure 16: Android Studio download

Again, it is required to click on the appropriate link depending on the user’s operating system to install Android Studio. The next step would be installing the Flutter and Dart plugins within Android Studio. After Android Studio is started, the user should select **Custom** as compared to **Standard** if given the choice. Make sure that the **Android Virtual Device** is selected to have access to an emulator for testing. Follow the remaining prompts and finish the installation of the selected items by default. After Android Studio is opened, the user should click on **Configure** at the bottom right and then **SDK Manager**. Go to **SDK Tools** and search for **Android SDK Command-line Tools** and make sure it is checked. Click **OK** and make sure the license is selected, click on **Accept** and then **Next→Finish**. Go to **Configure** again and click on **AVD Manager**. Click on **Create Virtual Device** at the bottom and choose an Android device of choice under **Phone** and install the emulator. **Pixel 3** is an example of the available choices and was used when testing the application. Choose an Android version, ideally the latest and click **Download**. Click on **Finish** and give your emulator a name after selecting it and clicking on **Next**. Finally, click on **Finish**. If on MAC, the user needs to go to **Preferences → Plugins**. From there, they must click on **Install** after they select the Flutter plugin. The user will be asked whether they want to install the Dart plugin as well and they should click **Yes**. Finally, they will be prompted with **Restart**, which they must click on to complete the process. On Windows or Linux, the user must go to **File → Settings → Plugins**. They

need to select **Marketplace**, after which they need to select on Flutter plugin and finally click on **Install**.

Now open a terminal by typing command prompt in the search location of your operating system. Click on the application to open it and type the following: **flutter doctor --android-licenses** as illustrated below:

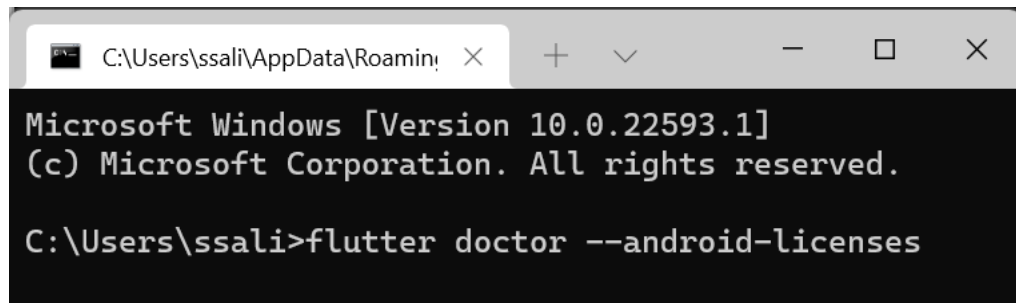
A screenshot of a Windows Command Prompt window. The title bar shows the path 'C:\Users\ssali\AppData\Roaming\...' and standard window controls. The command prompt text shows 'Microsoft Windows [Version 10.0.22593.1] (c) Microsoft Corporation. All rights reserved.' followed by the command 'C:\Users\ssali>flutter doctor --android-licenses'.

Figure 17: Running a command in the command prompt

When shown “Accept? (y/N):”, write ‘y’ without the quotation marks and press enter until the installation finishes.

A flutter project can now be created. Choose an appropriate location where the projects should be or alternatively, go your “Users” folder, then “username” and create a folder named “dev”. In it create a folder named “projects”. Back to the command prompt, make sure your path is “Users” followed by whatever your username is. Write while ignoring the quotation marks, “cd dev” and press enter. Then write “cd projects” and press enter again. Finally, write “flutter create ttsapp” where ttsapp is the project name which can be named whatever. However, it is highly recommended to only use lowercase letters and no special characters except for underscore “_” to separate between words. For example, my_project would be a valid project name, but not my-project.

Visual Studio Code (VS code) can also be installed at [Download Visual Studio Code - Mac, Linux, Windows](#). Like previously, the correct link will need to be clicked depending on the operating system the user is on. The user needs to then follow the installation prompts and launch VS code to start it. At the top left, they should go to **View → Extensions** and in the search bar type “flutter”:

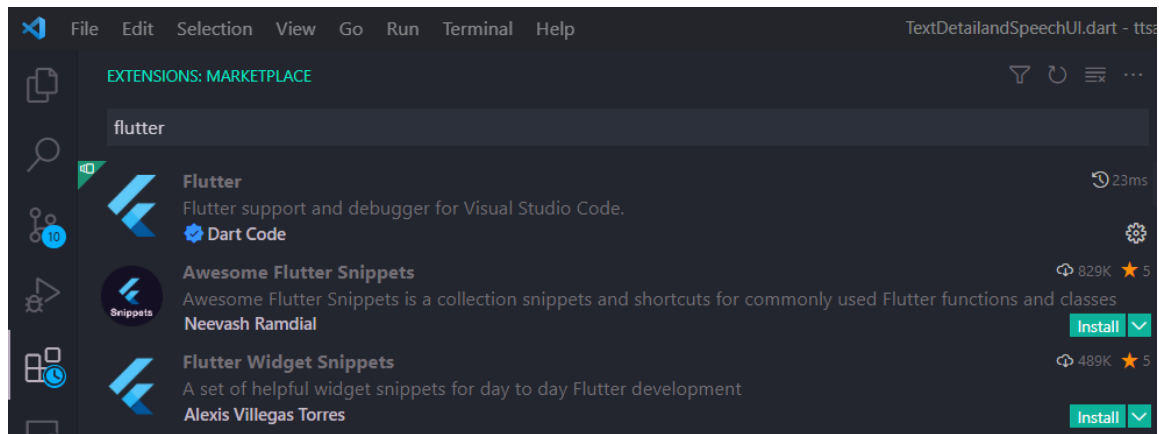


Figure 18: flutter extension installation on VS code

In my case, flutter is already installed, but the user should be prompted with an “Install” button at the right just like for “Awesome Flutter Snippets”. This extension provides great support when writing flutter code. This installation automatically installs the dart extension as can be seen “Dart Code” next to the blue check. Also search for Code Runner and install it in the same way. It is a user-friendly way of running your code.

Now to open up the created project, go to **File → Open Folder → Project → Open** where Project represents the path to your project. In this case, it would be `C:\Users\username\dev\projects\ttsapp` for example. Select **Yes, I trust the authors** if a dialog box opens. Now open Android Studio and go to AVD Manager to start the emulator:

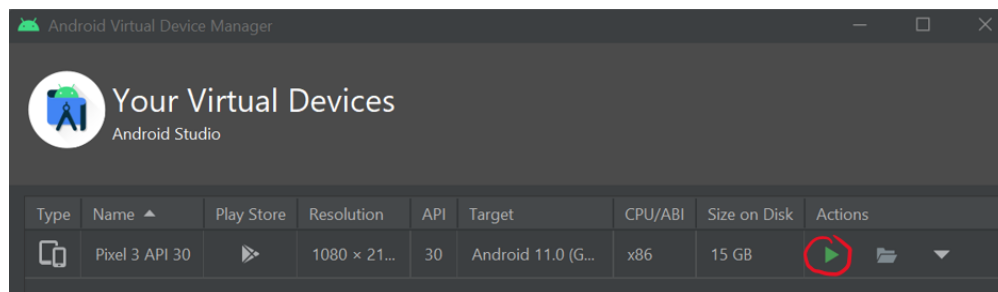


Figure 19: Starting the emulator

Click on the “Play” button in green circled in red in figure 13 to start the emulator:



Figure 20: Pixel 3 emulator

Now the different dependencies of the application must be installed. In VS Code, click on **Terminal** → **New Terminal** to open a new Terminal. Now run the following commands one after the other, by pressing “Enter” after each one:

- `flutter pub add image_picker`
- `flutter pub add file_picker`
- `flutter pub add flutter_tts`

The way the application is built is through an object-oriented manner and dynamically. The latter means that a change in some specific part of the code will not affect other parts and thus potentially causing some errors. There are a number of files included in this project and these should first be created. Click on the lib folder to create the following folders: Components, Provider, Routes, Services, UI and Utils.

```

Components
├── AppHeader
│   └── AppHeader.dart
├── Buttons
│   ├── FilledButton.dart
│   ├── OutlinedButton.dart
│   └── RadioButton.dart
├── CustomImages
│   └── NetworkImage.dart
├── Loaders
│   └── AppLoader.dart
├── showToast
│   └── showToast.dart
├── TextFields
│   ├── CustomTextField.dart
│   ├── CustomTextFieldWithBorder.dart
│   └── TextAreaField.dart
└── Widgets
    └── AppHome
        └── AppHome.dart

```

Figure 21: Project tree | lib folder part 1

```

main.dart
Model
└── onBoardingModel
    └── onBoardingModel.dart
Provider
├── ImageProvider
│   └── ImageProvider.dart
└── SelectFileDetailProvider
    └── SelectFileDetailProvider.dart
Routes
└── Routes.dart
Services
├── galleryOrCamerImageToText
│   └── galleryOrCameraImageToText.dart
├── SelectPdfToText
│   └── SelectPdfToText.dart
├── SharedPreferenceService
│   └── SharedPreferencesService.dart
├── SplashTimeService
│   └── SplashTimeService.dart
└── UI
    ├── AppHomeUI
    │   └── AppHomeUI.dart
    ├── onBoardingUI
    │   └── onBoardingScreen.dart
    ├── SplashUI
    │   └── SplashUI.dart
    └── TextDetailandSpeechUI
        └── TextDetailandSpeechUI.dart

```

Figure 22: Project tree | lib folder part 2

```

└── Utils
    ├── Constants
    │   ├── AssetConstants.dart
    │   ├── ColorConstants.dart
    │   ├── ConstantValue.dart
    │   ├── IconConstants.dart
    │   ├── KeysConstants.dart
    │   ├── NameConstants.dart
    │   └── RouteConstants.dart
    └── Globals.dart

```

Figure 23: Project tree | lib folder part 3

As can be seen above this is the structure of the application. The folders and files should be created in the appropriate place in order for VS code to be able to run the code accordingly. For Example Utils is a top level folder in the lib folder. In it, there is a Constants folder which has 7 files with the extension “.dart” as shown in the tree on Figure 17. The “Globals.dart” file is at the same level as the Constants folder. These files are all linked together through the “main.dart” file which is the brain of the application. That is, when the application runs, the “main.dart” file is the code that is executed. The link to the other files is done through the use of import statements at the top of the files. For example, if I want to use a color that has been defined in the “ColorConstants.dart” file in the “AppHeader.dart” file, I need to first import that file at the top of the “AppHeader.dart” file, and then use those colors by calling their appropriate names in the file. Errors have also been anticipated in case the file selected is not supported by the application for example. These errors are taken care in the “showToast.dart” file. All the files have been given a name that is self-explanatory for their intended use for easy navigation and good code organization. After all the code has been written in their appropriate files, an apk file can be generated by running a command in the terminal:

```
flutter build apk --build-name=1.0 --build-number=1
```

For deployment of the application in the Play Store or App Store, the documentation can be accessed at this link for the details: [Build and release an Android app | Flutter](#) and [Build and release an iOS app | Flutter](#).

6.2 Testing & Validation

Several test cases have been performed at every building stage of the application to make sure that the appropriate functionality was working correctly.

Test Case 1: Sanity test

1. Run the template code provided by flutter to make sure it runs properly.
2. Compare actual output to the desired output

Test Case 2: Image picker option

1. Install the image_picker plugin
2. Run the example code provided by flutter to make sure output is shown on emulator
3. Make appropriate changes to show the icon and perform action on click

Test Case 3: Camera access option

1. Install the image_picker plugin
2. Try available example codes to check the output
3. Make appropriate changes to add the icon and perform action on click

Test Case 4: File picker option

1. Install file_picker plugin
2. Run the available example code to check the output
3. Modify the code appropriately to add icons and perform action on click
4. Tested the exception thrown for unsupported filetypes

Test Case 5: Routes management

1. Tested the “page_transition.dart” package for transitioning between routes
2. Tested the transition type by using the built-in PageTransition function for duration and effect
3. Tested route exception in case some unexpected behavior occurs

Test Case 6: Image and text rendering

1. Installed the text_to_speech plugin
2. Tested available documentation and open-source examples to check the output
3. Tested the text extraction

Test Case 7: Voice output

1. Created and tested a function speak that runs on tap of “Read out loud button”
2. Tested multiple images and files to check any difference in behavior
3. Varied the speed multiple times to make sure that the voice is audible and understandable

Test Case 8: Splash Screen and Onboarding Screens

1. Tested the transition duration between the splash and onboarding screens
2. Tested whether the SKIP button brings the user to the main UI
3. Tested the if the Next button brings the user to the next onboarding screen

7 Conclusions and Recommendations for Future Work

To conclude, throughout the project, we learned about Flutter software development kit, and mobile app development. We also had the chance to enhance our coding skills. Our prototype was greatly enhanced after each client meeting to ensure that our client was satisfied. It was designed to be simple and easily navigable. In the future, other groups can show our app to their clients and demand them what modifications they can implement to optimize our app or make one of their own better than ours and those already in the market. If we had few more months to work on this project, we would have enhanced our application to have all the ideal values from our target specifications in order to make our application ideal. The things that we abandoned because of lack of time but would be important to add are all types of files to be supported, more voices to choose as the speaker, the ability to read more languages, iOS compatibility, more reading speed options, and a play/pause button so that the app can read from where it stopped.

8 Bibliography

A5. (2019, November 5). Image to speech. App Store. Retrieved January 24, 2022, from <https://apps.apple.com/us/app/image-to-speech/id1484770181>

“Downloads.” *Git*, <https://git-scm.com/downloads>.

Eddine, M. C. (2020, October 2). Talkie OCR - image to speech. App Store. Retrieved January 24, 2022, from <https://apps.apple.com/us/app/talkie-ocr-image-to-speech/id1512795289>

Schwarz Müller, Academind by Maximilian, and Maximilian Schwarz Müller. “Flutter & Dart - The Complete Guide [2022 Edition].” *Udemy*, Udemy, <https://www.udemy.com/course/learn-flutter-dart-to-build-ios-android-apps/>.

“Android Studio Tour - Flutter Video Tutorial: LinkedIn Learning, Formerly Lynda.com.” *LinkedIn*, <https://www.linkedin.com/learning/flutter-part-01-introduction/android-studio-tour?autoplay=true>.

Image_picker: Flutter Package. Dart packages. (2022, February 25). Retrieved March 6, 2022, from https://pub.dev/packages/image_picker

Flutter documentation. Flutter. (n.d.). Retrieved March 6, 2022, from <https://docs.flutter.dev/>

File_picker: Flutter Package. Dart packages. (2022, February 28). Retrieved March 6, 2022, from https://pub.dev/packages/file_picker

Flutter_tts: Flutter Package. Dart packages. (2021, December 24). Retrieved March 6, 2022, from https://pub.dev/packages/flutter_tts

Flutter text to speech (TTS) | flutter ... - youtube.com. (n.d.). Retrieved March 7, 2022, from <https://www.youtube.com/watch?v=IHA vWqp5chc>

Flutter text to speech (TTS) | flutter ... - youtube.com. (n.d.). Retrieved March 7, 2022, from <https://www.youtube.com/watch?v=IHA vWqp5chc>

Text to speech (TTS) in flutter | flutter tutorial - youtube. (n.d.). Retrieved March 7, 2022, from <https://www.youtube.com/watch?v=WnJZOi57oTY>

“Install.” *Flutter*, <https://docs.flutter.dev/get-started/install>.

Dhameliya, Milan. “[Solved] Error Running Pod Install in Flutter on Mac.” *Flutter Corner*, 29 Nov. 2021, <https://fluttercorner.com/error-running-pod-install-in-flutter-on-mac/>.

NitneugNitneug 3, et al. “How to Solve ‘Error Running Pod Install’ in Flutter on Mac?” *Stack Overflow*, 1 Oct. 1966, <https://stackoverflow.com/questions/54135078/how-to-solve-error-running-pod-install-in-flutter-on-mac>.

“Build and Release an Android App.” *Flutter*, <https://docs.flutter.dev/deployment/android#publishing-to-the-google-play-store>.

“Build and Release an IOS App.” *Flutter*, <https://docs.flutter.dev/deployment/ios>.

Microsoft. “Download Visual Studio Code - MAC, Linux, Windows.” *RSS*, Microsoft, 3 Nov. 2021, <https://code.visualstudio.com/Download>.

“Download Android Studio and SDK Tools : Android Developers.” *Android Developers*, <https://developer.android.com/studio>.

APPENDICES

9 APPENDIX I: Design Files

All design files were added in MakerRepo. The MakerRepo link to our project is:

<https://makerepo.com/ssmsourang/1111.gng2101c2p3image-descriptor>

Table 4. Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
Image picker - Flutter	https://pub.dev/packages/image_picker	Sep 1, 2021
File picker - Flutter	https://pub.dev/packages/file_picker	Mar 10, 2022
Text to speech - Flutter	https://pub.dev/packages/flutter_tts	Dec 24, 2021

10 APPENDIX II: Other Appendices

This section is not applicable