EasyReader: Text Reader for Visually Impaired People

Amirul Abdullah[1], Mohamad Amir Fahmi Mohd Rosli[2] and Norsaremah Salleh[3]
Computer Science Department, Kulliyyah of ICT, International Islamic University Malaysia, Kuala Lumpur, Malaysia
amirulabdullah979797@gmail.com, mohdamirfahmi96@gmail.com, norsaremah@iium.edu.my

Abstract— Text-to-speech helps many visually impaired people to read text easily. One of the commonly used tool is Google Text-to-Speech. It exists in most of the Android smartphones. However, it is just an engine to be used in another application such as google play books and google translate. It is limited to read certain books that are available in google play books. On google translate, we need to manually type the word in google translate application to convert it to voice. The method is not practical for visually impaired people. Hence, this study aims to develop a Text-to-Speech (TTS) application that visually impaired people can easily use. The system is made up with minimal number of easily spotted buttons and employed rather a simple step.

Keywords— Text-to-Speech, reading disabilities, Flutter development, assistive technology.

I. Introduction

Visual impairment refers to "a decreased ability to see to a degree that causes problems not fixable by usual means, such as glasses" [1]. Long sightedness or hypermetropia is the main cause of the visual impairment of people today. It is an eye disorder in which distant objects can be seen clearly, however objects that are closed in distance appeared blurred. The blurred effect is due to the incoming light being focused behind, instead of in front of the retinal wall. It affects over 13 percent of people aged between 20 and 20 and over 17 percent aged between 40 and 45. Long-sightedness people face difficulties in reading newspapers, online article, social media on the phone and other related activities that intensely use eyes within close distances [1].

Despite the availability of existing text-to-speech application, some of the applications are not flexible to be used, as they require the user to enter the word phrase themselves. Applications that are not user-friendly for example one with many hidden buttons and messy interfaces make it difficult for user to use and navigate [2].

In this study, we aim to develop a prototype application that would easily facilitate people who have difficulties in reading text from their phone screen. The prototype was developed using Flutter open source user interface (UI) software development kit and the programming language written in Dart [3]. The software enables user to scan the photo and convert it into text and sound. User can choose to upload photos or to take a picture using their phone camera. The prototype, however, is limited to only English language. The development of the prototype is expected to contribute to people who are visually impaired to enable them in reading text smoothly.

II. Related Work

There are a few similar applications that provide services to convert text to speech, or known as Text-to-Speech (TTS) apps. TTS usually made of two engines, which are Optical Character Recognition (OCR), and Test-to-Speech (TTS) engine. The function of OCR is to convert the image to text, while TTS convert the text to speech.

A. *FPT.AI (web-based app)*

FPT.AI is a web-based application that was developed to convert Vietnamese text into spoken words [4]. It was developed using Django for Python and in the form of web page, connected to the FPT.AI server via its application programming interface (API). The API acts as an interface between local host and remote FPT TTS server and it takes four arguments to form the Hypertext Transfer Protocol (HTTP) request before posting to the server.

The server will return a response to the host application for each button request. The response is in the form of JavaScript Object Notation (JSON) format and contains a static HTTP link to download the converted audio file into the mp3 format [4]. The result of the conversion (whether successful or not) will be displayed to user together with a message detailing the system's feedback regarding the request. Fig. 1 shows the main page of FPT.AI

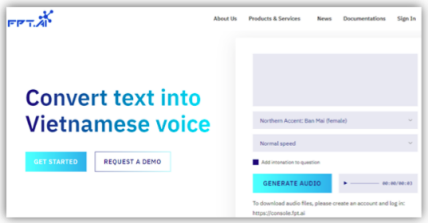

Fig. 1 Main page of FPT.Ai

B. @Voice Aloud Reader (Application)

@Voice Aloud Reader is a TTS application that can read text in the form of text, word doc and pdf [5]. It can be used as an extension to the browser, in which user visited news website using a browse can share it using the installed software. The full article will be automatically copied to the application and the computer speaker will automatically read the text and saved into a sound file in the form of Waveform Audio File (wav). This application however, consists of many buttons, icons and text, and therefore lack of user-friendly interface for visually impaired people. Fig. 2 showed the main page of @Voice Aloud Reader app.

Fig. 2 Main page of @Voice Aloud Reader

C. Raspberry Pi Text-To-Speech (Device)

Raspberry Pi Text-To-Speech used an inbuilt camera to capture the text image from the printed text and the image will be analysed by Tesseract-Optical Character Recognition (OCR) see Fig. 3. The detected text is then converted into speech using an open source software speech synthesizer, eSpeak. Raspberry Pi is used for interfacing between the camera, sensor and the image processing results [6].

The Tesseract OCR is built solely for the Text Information Extraction (TIE). TIE process is important to determine the intelligibility and the accuracy of the text conversion for the output speech. The processing within the Tesseract OCR follows the traditional steps. The first step is called Connected Component Analysis, where the outlines of the components are stored. This step is computationally intensive in which it could also read the reversed text. In the second step, the outlines and the regions analysed as blobs and the text lines are broken into character cells with algorithm nested in the OCR for spacing. The recognition phase is later divided into two parts and each word is passed to an adaptive classifier. Adaptive classifier is used for the OCR engine to decide whether the font is character or non-character. The extracted text will be saved in a text file, to be processed by TTS module, eSpeak. The eSpeak is an open source software that is used to synthesize the text into speech by converting the text file into an audio file (.wav).

Fig. 3 Raspberry Pi TTS device

Table 1 shows the comparison between several TTS solutions available in the market. Basically each of them has their own strength and weaknesses. For example, FPT.AI cannot be used in offline mode since it depends on its own website. Although @Voice Aloud Reader application has a multi-language converter, users need to download the desired language prior to using them. In this study, we developed EasyReader that is capable to scan images and convert it to text, however, it cannot scan document file.

Table 1. TTS Comparison

App tool/ feature	FPT.AI	@Voice Aloud Reader	Raspberry Pi	Easy Reader
Offline connection usage	□	□	□	□
Multi-language	□	□	□	□
Smartphone platform	□	□	□	□
Scan image to text	□	□	□	□
Read hand writing	□	□	□	□

Isewon et al. (2014) developed a text-to-Speech synthesizer that can convert inputted text into synthesized speech and reads out to the user. The synthesizer consists of two major modules: Natural Language Processing (NLP) module, which produced a phonetic description of the text read, and Digital Signal Processing (DSP) module for transforming the symbolic information from NLP into audible speech [7].

The effectiveness of TTS software intervention has been studied in Stodden et al. (2012) using two pilot studies involving 104 high school students [8]. Their findings indicated that the use of TTS software (for at least 30 minutes per week) had significantly improved students' reading skills. Similar findings were obtained in a meta-

analysis study reported by Wood et al. (2018). They mentioned that the text-to-speech technologies would assist students in reading comprehension. However, suggesting that more studies are required to explore the effect of the moderating variables of text-to-speech and read aloud tools' effectiveness for reading comprehension [9].

III. Methodology And Design

A system development methodology refers to the framework that is used to structure, plan, and control the process developing a system [10]. In this study, we employed Software Prototyping development methodology that consists of communication, construct prototype, customer evaluation, iterate prototype, and deploy software [10]. The aim is to reduce the risk and cost while focusing on quality and performances of the system.

A. Architectural Design

Requirements for the EasyReader application are elicited in the communication process. At this stage, the main features expected for the system are discussed. Fig. 4 shows the main use-cases for EasyReader. The architecture of EasyReader can be respresented using the 4+1 view model architecture [11].

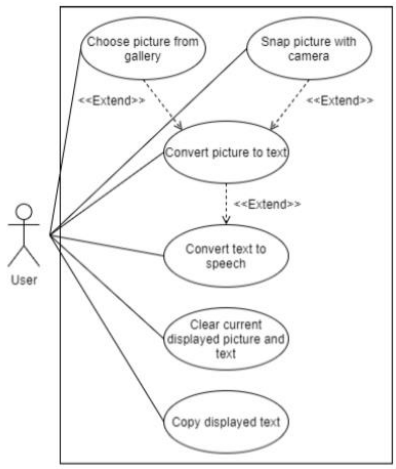

Fig. 4 Use case diagram

The logical view of the system is represented using the class diagram (see Fig. 5). It consists of four classes called *User*, *Image*, *Speech*, and *Text*. To model the static implementation view of EasyReader, we used a component diagram (see Fig. 6) to illustrate how the physical subsystems are organized according to its hierarchy. For instance, Firebase ML Vision is the main package that consists of cloud-enabled API plugins for handling images [12]. Another important component is known as Flutter TTS which responsible to provide plugins for converting text to speech.

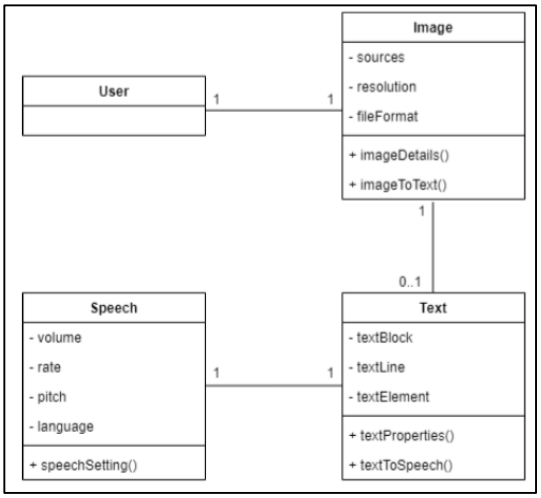

Fig. 5 Class Diagram

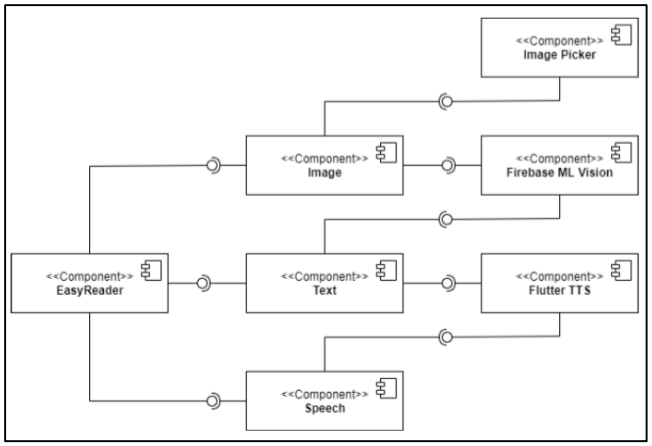

Fig. 6 Component Diagram

Finally, to model

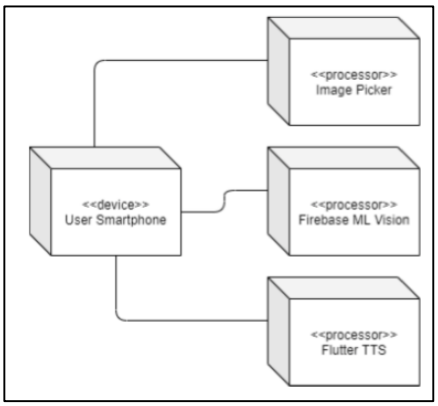

Fig. 7 Deployment Diagram

B. Application Programming Interfaces and Packages

The Application programming Interface (API) used in this study is Firebase Machine Learning Vision or known as Firebase ML Vision [12]. In Flutter, this API is called a package that provides features such as barcode scanner, face scanner, image labeller, text recognizer and document text recognizer. The API converts the image captured into text.

The Flutter TTS package is used to convert the text to speech. This package provides important features such as play, pause, stop, language preferred, voice pitch, loudness and rate.

C. *Prototype Construction*

In constructing the prototype, we started with low fidelity prototype to enable the system to capture basic requirements needed by the user [10]. When the system design has been detailed out, we migrated to high fidelity prototype. The development of the prototype is to be prioritized, as we wanted to maximize users' needs and satisfaction.

The design for the input data is intended to make the data entered easy as possible and to minimize human error. In our system, the user input comprises the picture uploaded from a gallery or a picture taken using the phone camera. Sample of the code used for recognizing the text from the image is shown in Appendix A.

IV. Analysis of Results

The results from this study produced a system known as EasyReader to be used mainly by visually impaired people to assist them in reading the text-image. EasyReader allows user to read the text from any image taken either from the gallery or snapping from the phone camera. The text-image output will be shown on the screen before scanning process took place. When the user click on the "scan button", the text from the image will be shown on the screen below the displayed image. Then, the user can click the speaker icon to make the system read the displayed text and the user will listen accordingly.

The system produces two types of output, which are text and speech. The quality of the output significantly depends on the picture uploaded into the system. For the text part, if the text extracted is lengthy, user needs to scroll down the text box area. If the image produced is without text, the text box will display "no text found in the image" as a feedback to the user.

For the speech part, the system's narrator will say out loud that there is no text found if the image did not provide any text. Images that contain text will be read out loud by the narrator voice accordingly. The narrator voice cannot be paused or stop as these features are not yet available in the Flutter development kit.

Fig. 8 showed the main page of EasyReader. It consists of four buttons and two clickable icons. The buttons are "Gallery", "Camera", "Scan", and "Clear". The icons are "Copy" and "Speaker". The copy button will copy the text displayed and the speaker button will play the text voice.

Fig. 8 Main page of EasyReader

Once the user click the "Gallery" button, the system will show the phone's gallery and the user can choose the picture accordingly (see Fig. 9). If the user clicks "Camera" button, the camera module will be opened for the user to snap pictures when they need to do so. Once the user chose the preferred text-image, the text image will be displayed on the screen. User can click on the "Scan" button to display the scanned text in the text box (see Fig. 10).

Fig. 9 Displaying photo gallery

Fig. 10 Displaying the scanned text

To ensure the system's functionalities work effectively, system testing was performed using several test cases. During the testing, few potential inputs were identified and the results showed that the system can capture the image and translated to audio voice successfully. Sample of the test script is shown in Fig. 11.

Test Case ID	F003	**Test Priority**	High
Test Designed by	Amirul	**Date of Test Design**	16/01/2021

Test File	Scan the displayed text-image
Description of Test	This test will identify whether the scanned text can be displayed or not
Pre-condition	The text-image has been displayed on the screen
Test Steps	1. Click 'Scan' button to scan the text from the text-image
Expected Results	The scanned text will display on the screen
Status	Pass / Fail
Comment	

Fig 11. Sample Test Script

V. CONCLUSIONS

This study aims to design and develop a prototype application that can assist users who are visually impaired to read text-image easily. The prototype, known as EasyReader is designed as a mobile application that can be connected to the Firebase ML Vision to scan the text-image and convert it to text. In the prototype development, the team uses Flutter open source UI software development kit, which uses Dart as a main programming language. Visual Studio code is used as a compiler and Android Studio is used as an emulator. When the prototype is completely built, we installed EasyReader into a real android device for testing purposes.

Overall, the prototype has been designed with simple interfaces and minimal use of buttons so that the users can use it easily. For future work, the prototype can be improved by considering text-image in other languages. Additionally, feature such as language translator can also be included to address the current limitation of the prototype.

ACKNOWLEDGMENT

We would like to thank the participants involved in the testing and evaluation of EasyReader.

REFERENCES

[1] Understanding Vision Loss. The Outreach Center for Deadness and Blindess (2020) [online]. Available: https://deafandblind.outreach.org/understanding-vision-loss

[2] L. Punchoojit & N. Hongwarittorn, "Usability studies on Mobile user interfaces: A systematic literature review", *Advances in Human Computer Interaction*, vol 2017, Article ID 6787504, 2017, https://doi.org/10.1155/2017/6787504

[3] Fullter (2020) [online]. Available:https:///flutter.dev

[4] T.D. Chung, M. Drieberg, M.F. Hassan & A. Khalyasmaa, "End-to-end conversion speed analysis of an FPT.AI based Text-to-Speech Application", in Proc. IEEE 2nd Global Conference on Life Sciences and Technologies, 2020, pp. 136 – 139.

[5] Void Aloud Reader (2020) [Online]. Available: https://voice-aloud-reader.en.uptodown.com/android

[6] H. Rithika & B. N. Santoshi, "Image text to speech conversion in the desired language by translating with Raspberry Pi", in Proc. IEEE International Conference on Computational Intelligence and Computing Research (ICCIC2016), 2016, pp 3 – 6.

[7] I. Isewon, J. Oyelade, O. Oladipupo, "Design and Implementation of Text to Speech Conversion for Visually Impaired People", *International Journal of Applied Information System*, vol. 7, no. 2, pp. 25 – 30, 2014.

[8] R.A.Stodden, K.D. Roberts, K. Takahashi, H.J. Park and N.J. Stodden, "Use of Text-to-Speech Software to improve reading skills of High School Struggling Readers", *Procedia Computer Science*, Vol. 14, pp. 359 – 362, 2012.

[9] S.G. Wood, J.H. Moxley, E.L. Tighe, and R.K. Wagner, "Does use of Text-to-Speech and Related Read-Aloud tools improve reading comprehension for students with reading disabilities? A meta-analysis", Journal of Learning Disability, 2017. doi: 10.1177/0022219416688170

[10] A. Susan & Meryani, "System Development Method with the Prototype Method", International Journal of Scientific and Technology Research, vol 8 (7), 2019, pp 141 – 144.

[11] 4+1 Architectural View Model – CodeOpinion. (n.d). Retrieved December 31, 2020, from https://codeopinion.com/41-architectural-view-model/

[12] Firebase_ml_vision|Flutter Package. (n.d). Retrieved December 31, 2020, from https://pubb.dev/packages/firebase_ml_vision

APPENDIX A

```dart
import 'dart:io';

import 'package:firebase_ml_vision/firebase_ml_vision.dart';

class FirebaseMLApi {
  static Future<String> recogniseText(File imageFile) async {
    if (imageFile == null) {
      return 'No selected image';
    } else {
      final visionImage = FirebaseVisionImage.fromFile(imageFile);
      final textRecognizer = FirebaseVision.instance.textRecognizer();
      try {
        final visionText = await textRecognizer.processImage(visionImage
);
        await textRecognizer.close();

        final text = extractText(visionText);
        return text.isEmpty ? 'No text found in the image' : text;
      } catch (error) {
        return error.toString();
      }
    }
  }

  static extractText(VisionText visionText) {
    String text = '';

    for (TextBlock block in visionText.blocks) {
      for (TextLine line in block.lines) {
        for (TextElement word in line.elements) {
          text = text + word.text + ' ';
        }
        text = text + '\n';
      }
    }

    return text;
  }
}
```