

# **Android Based Sinhala Document Reader for Visually Impaired People**

**A.K.P.D. Mishangi**

**2018**



# **Android Based Sinhala Document Reader for Visually Impaired People**

**A dissertation submitted for the Degree of Master of  
Computer Science**

**A.K.P.D. Mishangi**

**University of Colombo School of Computing**

**2018**



## **Declaration**

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: A.K.P.D. Mishangi

Registration Number: 2015/MCS/047

Index Number: 15440472

---

Signature:

Date:

This is to certify that this thesis is based on the work of

Ms. A.K.P.D. Mishangi

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Dr. A.R. Weerasinghe

---

Signature:

Date:

## **Abstract**

In this era knowledge is the key factor for successful survival. Knowledge is gained through many ways and reading is one of them. For a person who is visually impaired, reading can be performed through braille system. In today's context, this traditional braille systems are replaced with computer aided braille systems or text to speech systems. Though these technologies are widely available for English, their availability is rare for Sinhala. Also such sophisticated systems are highly priced so that an average Sri Lankan cannot afford them. Since mobile phones play a significant role in people's life and when it comes to Sri Lanka, Android is the market leader for mobile operating systems, an Android based solution was ideal for this research problem.

This research solution detects the document and automatically captures it. Then passes it to the Tesseract optical character recognition engine, which identifies text if the relevant dataset files are given and passes text output to Android's text to speech system which reads out text input for the user using device's default TTS engine. Also by providing necessary audio guidelines and notifications, this application makes the life easier for its users. Graphical user interface of the application is simple and designed in a way which is user friendly for visual impaired community.

Functional evaluations were carried out to assess viability of the main three sub systems. Document detection accuracy is recorded as 100 percent on tested document sizes, provided document's color and background color contrast is high. Character recognition accuracy is evaluated by comparing Sinhala and English languages and OCR's accuracy is really low for Sinhala compared to English. Text to Speech system's accuracy depends on OCR's output. If accurate text inputs were given, TTS can give 100 percent accurate audio output. As a non-functional evaluation a usability evaluation was carried out which has scored around 90 percent positive feedback overall. Thus, this application is considered a valuable application for the visually impaired people in Sri Lanka to fulfill their craving for knowledge due to its usability and capabilities.

## **Acknowledgements**

Firstly, I would like to convey my sincere gratitude to my supervisor Dr. Ruvan Weerasinghe for his continuous support, motivation, knowledge sharing, patience and guidance throughout this research study.

I would like to thank all UCSC lecturers and staff for all the support they have given to me throughout this period.

Finally, I would like to take this opportunity to express my heart felt gratitude to my loving parents, my sister and my friends for the immense support they gave throughout this time to make this project a success.

# Table of Contents

List of Figures.....	vii
List of Tables.....	viii
List of Abbreviations.....	ix
1. Introduction.....	1
1.1 Research Objective.....	2
1.2 Scope.....	2
1.3 Outline.....	2
2. Literature Review.....	3
2.1 Optical Character Recognition.....	3
2.2 Text to Speech.....	4
2.2.1 Research on Available Text to Speech Systems.....	5
2.3 Android Based Related Work.....	6
3. Methodology.....	9
3.1 Android Application.....	9
3.2 Functional Requirements.....	10
3.3 Non-Functional Requirements.....	11
3.4 Design.....	12
3.4.1 Document Detection Module.....	12
3.4.2 Skew Detection and Correction Module.....	12
3.4.3 Optical Character Recognition Module.....	12
3.4.4 Text to Speech Module.....	12
4. Implementation.....	15
4.1 Design Decisions.....	15
4.1.1 Application for Smartphones.....	15
4.1.2 Android Technology.....	15

4.2 Development Environment.....	17
4.2.1 Operating System .....	17
4.2.2 Android Studio IDE.....	17
4.2.3 Source Code Repository .....	17
4.3 Technologies.....	17
4.3.1 Android Software Development Kit.....	17
4.3.2 OpenCV Library .....	19
4.3.3 Tesseract OCR.....	19
4.3.4 Android TTS.....	20
4.4 System Architecture .....	20
4.5 Execution Flow of Application .....	21
4.6 Key Functions.....	21
4.6.1 Document Detection .....	21
4.6.2 Character Recognition .....	23
4.6.3 Text to Speech .....	24
5. Evaluation and Results .....	25
5.1 Functional Evaluation.....	25
5.1.1 Document Detection Accuracy.....	25
5.1.2 Optical Character Recognition Accuracy .....	28
5.1.3 Text to Speech Accuracy .....	30
5.2 Non Functional Evaluation.....	31
5.2.1 Usability Evaluation .....	31
6. Conclusion and Future Work.....	33
6.1 Research Conclusion .....	33
6.2 Future Work.....	33
References .....	35

A. Application Design Diagrams .....	38
A.1 Use Case Diagram of the Application .....	38
A.2 Sequence Diagram of the Application .....	39



## List of Figures

Figure 2-1: Overview of an OCR system .....	4
Figure 2-2: Overview of a typical TTS system .....	5
Figure 3-1: Application overview diagram .....	9
Figure 3-2: Flow chart of the application .....	13
Figure 4-1: Android Architecture .....	16
Figure 4-2: Android Activity life cycle .....	18
Figure 4-3: Architecture of Tesseract OCR engine .....	19
Figure 4-4: Architecture Diagram .....	20
Figure 4-5: Skew correction algorithm.....	21
Figure 4-6: Border detection algorithm .....	22
Figure 4-7: Initializing OCR engine .....	23
Figure 4-8: Initializing TTS.....	24
Figure 5-1: Real time document detection .....	26
Figure 5-2: High accuracy in document detection.....	26
Figure 5-3: Importance of background color contrast in document detection.....	27
Figure 5-4: Comparison of Sinhala and English OCR function.....	28
Figure 5-5: Questionnaire format .....	31
Figure A-1: Use case diagram of the system .....	38
Figure A-2: Sequence diagram for the system .....	39

## List of Tables

Table 2-1: Comparison of commercially available similar applications .....	8
Table 3-1: Functional requirements of the system .....	10
Table 3-2: Non-functional requirements of the system .....	11
Table 5-1: Tested devices .....	25
Table 5-2: Tested Documents .....	27
Table 5-3: English OCR output .....	29
Table 5-4: Sinhala OCR output .....	30
Table 5-5: Test users' feedback summary .....	32

## List of Abbreviations

OCR Optical Character Recognition

TTS Text to Speech

API Application Programming Interface

OS Operating System

IDE Integrated Development Environment

APK Android Package

SDK Software Development Kit

# Chapter 1

## 1. Introduction

Visually impaired people are all around the world. According to World Health Organization statistics 285 million people are estimated to be visually impaired worldwide. Out of which 39 million are blind and others have low vision. Also out of visually impaired people, approximately around 90% of visually impaired people live in developing countries under low income settings. From that percentage Sri Lanka is also a country with lot of people who are visually impaired [1].

In today's context, knowledge has become the key factor for successful survival. Reading becomes the main method of gaining knowledge. The visually impaired people are interested in gaining information and knowledge but the problem is that they cannot find every source of knowledge in braille format, because many sources of information are not in the format to support visually impaired people. Many technologies have answered this by implementing computer aided braille systems and text to speech systems. Most of these systems are widely available in developed countries and these systems are of high price. And when it comes to text to speech system they are mostly for English language. But for Sri Lankans whose mother tongue is Sinhala and whose English literacy is low aren't blessed with such facilities. Therefore we are in need of such a system.

The solution should be affordable for an average income earning person in Sri Lanka. Because if the solution is more costly to afford that will not make much effect in Sri Lankan context. The solution should also be convenient to in users' perspective and should be portable to add more value to it and to match with lifestyle of users in today's context.

We are in the era of technology and moving into the mobile based technologies at a rapid pace worldwide. Though Sri Lanka is still a developing country if we consider the overall usage of mobile devices, almost each and every citizen in Sri Lanka is an owner of a smartphone. Globally and also within our country Android has dominated the mobile phone industry. Therefore, an Android based mobile approach is considered as the ideal choice to solve this problem.

## **1.1 Research Objective**

To implement a mobile based Sinhala document reader to support visually impaired Sri Lankans to fulfil their craving for information and knowledge gaining. With this application the visually impaired people will be able to capture a document with the camera on their mobile device and listen to the document's content comfortably with lesser effort.

## **1.2 Scope**

- In this application a document must be detected, it should be captured automatically, Sinhala text on it will be detected and the text will be read out aloud.
- Accuracy of text recognition depends on Google Tesseract Optical Character Recognition (OCR) engine.
- Accuracy of the Text to Speech system depends on the available TTS engine within device and the text input received from OCR engine.
- This application does not support hand written text.
- This application does not support font sizes below 12-point font size.
- Proposed solution is only available for Android devices at least with 5 mega pixel camera and auto focus functionality.

## **1.3 Outline**

The remainder of this report is structured as follows: Chapter 2 gives literature review; Chapter 3 provides methodology and the design; Chapter 4 gives implementation details of the application; Chapter 5 provides evaluation and results and Chapter 6 gives conclusion and future work.

## Chapter 2

### 2. Literature Review

This chapter covers the related work in following areas: Optical Character Recognition (OCR), Text to Speech (TTS) and Android based related work.

#### 2.1 Optical Character Recognition

Optical Character Recognition is a mechanism which reads and converts printed, typed or scanned handwritten documents in to a computer readable format. OCR is the commonly used method to digitize the printed documents. In that way they can be used efficiently by storing in the electronic format where users can access, edit and search important information conveniently in timely manner. This mechanism commonly has usages in digitizing old documents, text to speech systems, text mining applications, forensic applications and surveillance applications [2].

All OCR systems include an optical scanner to read text and software tools to analyze and recognize texts in the images. Advanced OCR systems supports different text sizes and lot of languages around the world. But even such advanced systems find it difficult to deal with handwritten text recognition. OCR is an area of research in fields as artificial intelligence, computer vision and pattern recognition [3].

In the area of OCR with respect to mobile computing specially in Android platform, Tesseract is considered to be the freely available OCR engine with higher accuracy level. Tesseract engine was developed at Hewlett Packard (HP) labs between 1985 and 1994. In 2005 this software was made available as open source and from 2006 onwards Google has sponsored the development of this software [4].

Pre-processing, character recognition and post-processing are the techniques used in OCR engines like Tesseract. When performing character recognition, software like Tesseract uses a two-pass approach to enhance quality of output when characters in input image are distorted due to low quality scanning. The second pass known as “Adaptive recognition” is used to recognize the remaining letters of the first pass output [5]. Following figure demonstrates the overview of a typical OCR system [6].

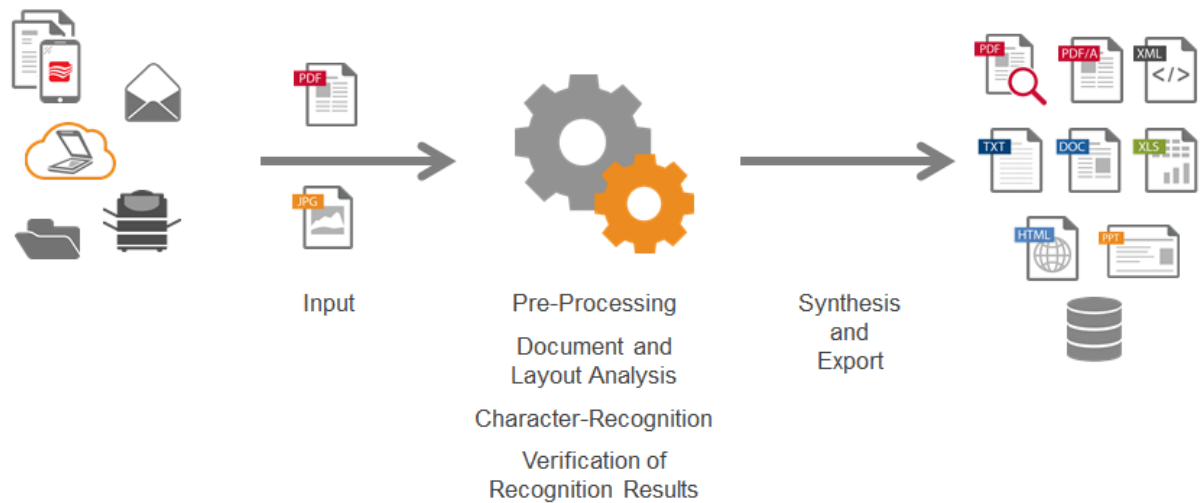


Figure 2-1: Overview of an OCR system

## 2.2 Text to Speech

Speech Synthesis is the process of producing human speech artificially. Speech Synthesizer is the computer based system that produces artificial human voices. Synthesize speech is created by the concatenating process of recorded speech files that are stored in the database. Text to Speech converts language text into speech whereas other systems convert representations of symbolic linguistics into speech. Concatenation and Formant are the two technologies in Speech Synthesis. Concatenation synthesis is based on concatenation of recorded speech segments and main sub types are Unit Selection and Diphone synthesis [7].

A text-to-speech (TTS) system is capable of making an audible output with a digital text input. An intelligible TTS system is much important to visually impaired people and people who are not literate. Though these systems were first designed and developed to cater the visually impaired community, nowadays they are integrated for much broader applications such as virtual assistants, public announcement systems and for voice based navigation systems [7].

TTS is comprised of two major parts, front end and back end. Text normalization and pre-processing are the two major tasks done by front end. Then it assigns phonetic transcriptions to each word, this process is called as text-to-phoneme conversion and then marks the texts into prosodic units like phrases, sentences and clauses. Prosody information and phonetic transcriptions make up symbolic linguistic representation which is the output of front end. The back end known as Synthesizer then converts front end output to sound. Following figure illustrates the overview of a typical TTS system [7].

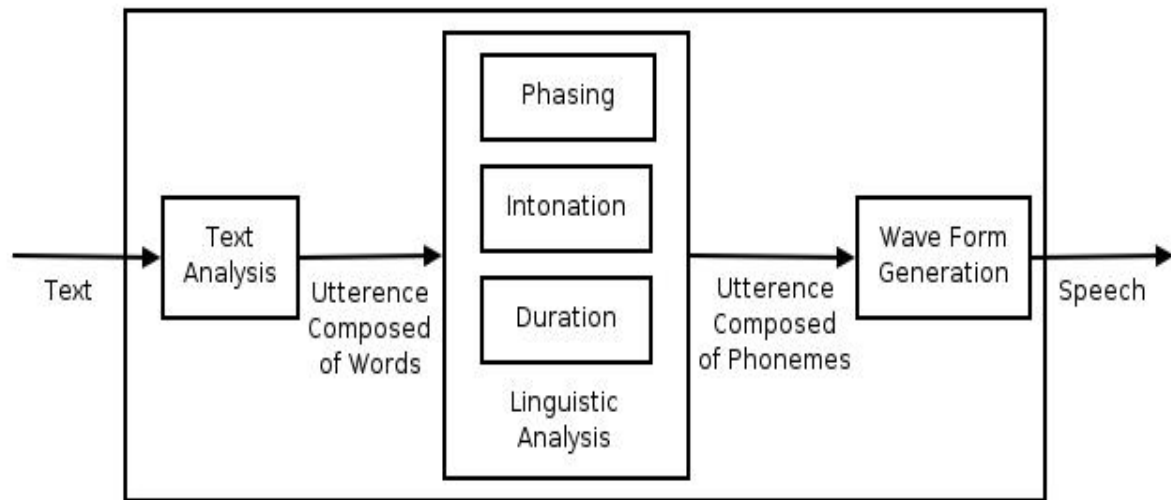


Figure 2-2: Overview of a typical TTS system

## 2.2.1 Research on Available Text to Speech Systems

TTS systems are widely available for English Language and their performance is much better compared to TTS systems developed in other languages. Therefore apart from Sinhala TTS systems available the English TTS systems were studied as well in order to get a better knowledge in the area.

### 2.2.1.1. Festival

Festival is the most complete, multilingual, stable and open source synthesis system which is originally developed by Alan W. Black at Centre for Speech Technology Research (CSTR) at the University of Edinburgh. This is written in C++ language and this offers a full TTS system with various APIs. Also this provides an environment for research and development of speech synthesis systems. For Festival a general purpose unit selection engine called ‘multisyn’ is used [8].

### 2.2.1.2. Festival-Si

This is based on Festival architecture and is the first ever documented work of a Sinhala TTS system developed by Language Technology Research Laboratory (LTRL) of University of Colombo School of Computing (UCSC). Development of a diphone database and the natural language processing modules were done. This is a computer based application which is developed to cater the visually impaired people in Sri Lanka to work on their computers using their local language [9].



### 2.2.1.3. Ivona

This is a multi-language TTS developed by a Polish software company known as IVO which was later acquired by Amazon. This system supports 22 languages in 51 different voices as of July 2017. Compared with other systems this supports many platforms including Windows, Android, iOS, Tizen and Unix based systems. This uses a Unit Selection algorithm with Limited Time-scale Modifications (USLTM) to provide full text-to-speech system with various APIs. This is one of the best commercially available paid TTS software tool available in the market [10].

### 2.2.1.4. eSpeak

An open source text-to-speech system which is developed by Free Software Foundation Inc. This was originally named as ‘speak’. The currently available version which is named as ‘eSpeak’ is an enhancement to ‘speak’ with features such as new language support, better handling of processing power constraints and relaxation of original memory. This is for Linux, Windows and other platforms. This supports Sinhala TTS functionality as well [11].

### 2.2.1.5. AMoRA

A TTS system for Sinhala Unicode which is based on diphone developed by University of Moratuwa. This project is also on top of Festival framework. This introduced a new phrase breaking algorithm using Sinhala linguistic features with an end user application and a new Sinhala based female was developed [12].

Considering the mobile platform Android has their own text-to-speech engine that supports Android version 1.6 upwards. Considering iOS platform from iOS7 Apple started offering API support for text-to-speech [12].

## 2.3 Android Based Related Work

An OCR project was implemented by a previous Master Degree student of UCSC to recognize Sinhala characters real time and to translate that input into English. This was aimed to help foreigners to understand Sinhala based notices and sign boards who don’t have knowledge in Sinhala language. This was one using Google Tesseract OCR engine. Another Master Degree student implemented an augmented reality based Tamil to Sinhala translator. This project also utilized the Google’s Tesseract as the OCR engine [13], [14].

There are commercially available software in Google Play which have OCR and TTS functionalities. But those applications are not for people who are visually impaired. One such widely used application is “Google Translate”. This application is available for free in Google Play. User can give typed or handwritten text, images and speech as input then it translates input to another language. Then user can hear the speech file of that translated input in second language [15].

“Google Translate” application supports only handwritten or typed input in Sinhala and doesn’t offer speech functionality for Sinhala. Google Translate provides offline support for some of the languages but doesn’t support for Sinhala offline translation. This application doesn’t have border detection mechanisms. This application is available in interfaces like web and iOS. Also API is available for developers to build other software and browser extensions [15].

Another application done by Google and freely available in Play Store is “Google Goggles”. This was developed as an image searching engine. But this performs the OCR functionality and make it easy for user to search through internet by using the text output identified by OCR. This application doesn’t support border detection and doesn’t recognize Sinhala texts [15].

“Text Fairy” is another OCR which supports many languages including Sinhala. But its accuracy level is really low when it comes to Sinhala language. And this application doesn’t do any border detection and lot of intermediate input from user is needed. But this work quite good with English OCR functionality, therefore it is a top rated and mostly used application in Google Play Store [15].

Another famous mobile scanner application is “Office Lens”. This scans documents, whiteboards, images and business cards. This saves the scanned files in internal memory of the device. When saving into internal memory application allows user to save the scanned documents in variety of formats such as a Microsoft Word document, a Power Point presentation, a Portable Document Format (PDF) file or as an image. In this application also OCR functionality works only for English language. This application has a good border detection mechanism and English OCR functionality is well accurate compared to other applications [15]. Comparison for aforementioned commercially available applications are in Table 2-1.

Name	Available Sinhala OCR function	Available English OCR function	Available Sinhala TTS function	Available English TTS function	Notes
Google Translate	√	√		√	Can give input as typed or handwritten text, images and speech. Supports many languages.
Google Goggles		√			Can give images as input. No automatic border detection mechanisms.
Text Fairy	√	√			Supports many languages. Sinhala OCR accuracy is low compared to English.
Office Lens		√			Good border detection and high accuracy in English OCR.

Table 2-1: Comparison of commercially available similar applications

When it comes to Sinhala OCR and TTS functionality there is no such applications to cater the need of Sri Lankans in area of OCR and TTS. And there is no application that is commercially available for visually impaired people in Sri Lanka with more accessibility functions enabled and with higher user experience enhancing functionalities. This research is aimed for the later purpose, to cater the visually impaired people in Sri Lanka to gain knowledge through listening to the content of reading materials printed in Sinhala language.

## Chapter 3

### 3. Methodology

This chapter covers functional and non-functional requirements of the proposed system. Also covers detailed application design information and the detailed methodological approach of the research.

#### 3.1 Android Application

As discussed in Chapter 1, this research aims to develop a solution to cater the visually impaired people in Sri Lanka to read books and papers printed in Sinhala. A mobile application in Android platform is designed to demonstrate the proposed solution. It can capture a document using camera in mobile device and extract Sinhala text in it and read it aloud for user in Sinhala. Following figure shows the overview of the proposed solution.

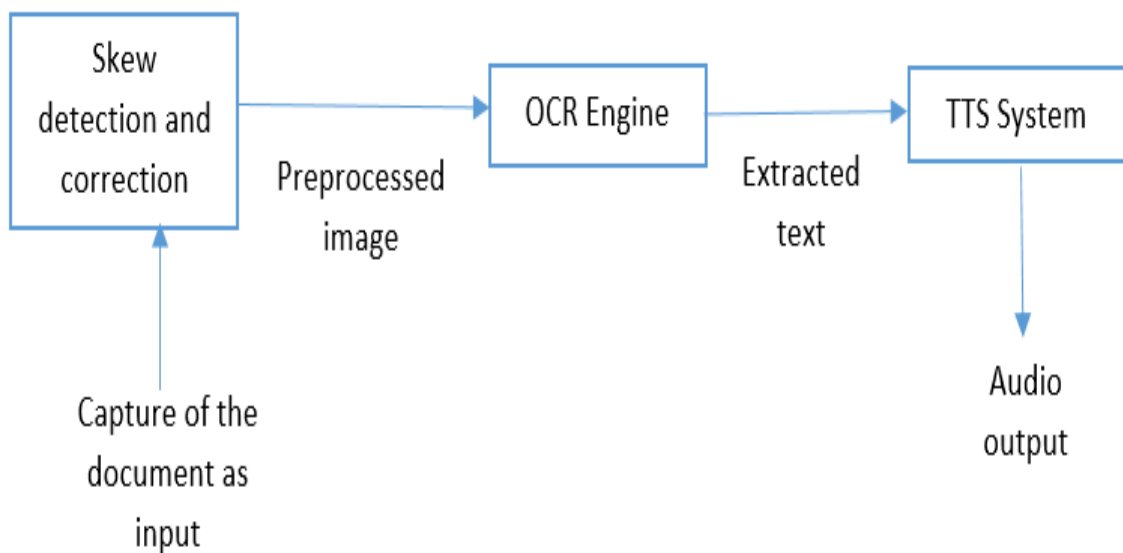


Figure 3-1: Application overview diagram

## 3.2 Functional Requirements

Following are the detailed functional requirements expected from this system.

<b>ID</b>	<b>Description</b>
1	Application should be able to start camera when user opens the application.
2	Application should be able to scan the document in real time when user starts the application.
3	Application should be able to autofocus the document in front of camera.
4	System should guide user with audio notifications until document is inside the capture frame.
5	Once the document is fully inside the capture frame system should be able to capture it automatically and retrieve the image.
6	Captured image should be saved to the device storage.
7	System should detect skew and correct skew before sending it to the OCR system.
8	OCR should be able to process the transferred skew corrected image to recognize Sinhala text within it.
9	User should be informed once the OCR process is done via an audio notification.
10	Extracted and recognized Sinhala texts should be transferred to the TTS system.
11	TTS should be read out the Sinhala text detected from the image captured by the camera.

Table 3-1: Functional requirements of the system

### 3.3 Non-Functional Requirements

Following are the detailed non-functional requirements of the system.

<b>ID</b>	<b>Description</b>
1	System should consist of convenient Graphical User Interface which uses Android usability functionalities.
2	Application should comply with Android architecture.
3	Application should comply with existing memory architecture of Android devices.
4	System should guide user with meaningful and various audio notifications for different functionalities.
5	Document use as input resource should be a printed material. Handwritten text documents are not supported.
6	User should hold the device steady when capturing the document.
7	System should place action buttons in the user interface in convenient locations and should provide audio based notifications upon selection of functionalities so user can conveniently use them.

Table 3-2: Non-functional requirements of the system

## **3.4 Design**

The application consists of the following core modules. Detailed descriptions for them is given in this section.

- Document detection module
- Skew detection and correction module
- Optical character recognition module
- Text to speech module

### **3.4.1 Document Detection Module**

Document detection is a critical factor in this application as the target audience is visually impaired and also without document detection system will not be able to capture the full document which will lead to inaccurate results in next modules.

OpenCV (Open Source Computer Vision) is a programming library designed for real time computer vision. This library was initially created in Intel Research labs and now it is maintained by Itseez. The core of the library is written in C++ programming language with Python interface. Many wrappers are available for other programming languages. JavaCV is the wrapper for Java language. JavaCV doesn't provide just one-to-one wrapper around OpenCV, it is a bundle of other image processing libraries like FFmpeg and OpenKinect. But within this application OpenCV is used for this purpose as it is an already evolved solution compared to JavaCV.

### **3.4.2 Skew Detection and Correction Module**

Once the image boundaries are detected it is automatically captured and sent for detection and correction of skew. Again the OpenCV library was chosen to achieve this functionality as it has skew correction functionality as well.

### **3.4.3 Optical Character Recognition Module**

The application passes the skew corrected image to the OCR for recognition of Sinhala characters which is on scanned document. To use as the OCR system, most evolved and mostly used Tesseract engine was chosen.

### **3.4.4 Text to Speech Module**

This module get recognized Sinhala text phrases as the input. And this will output relevant audio files for the application. Google TTS functionality was chosen as the default TTS engine for this application.

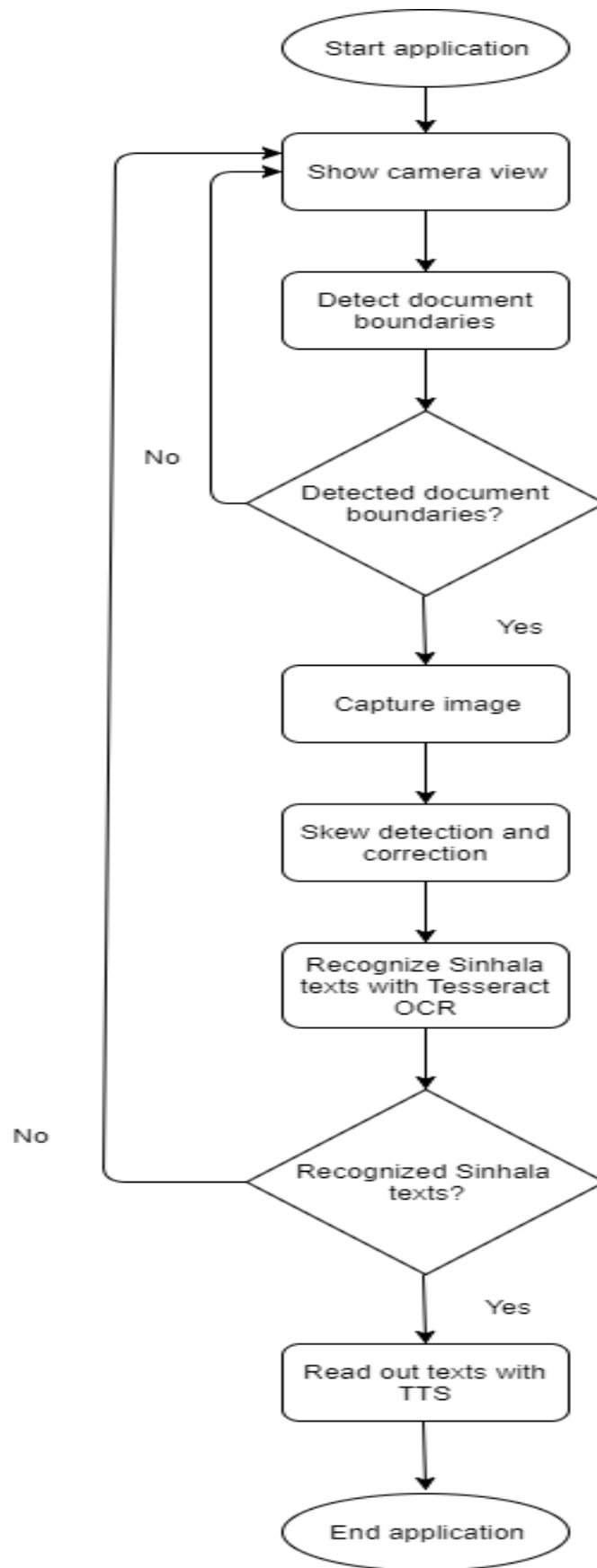


Figure 3-2: Flow chart of the application



Figure 3-2 illustrates the process flow of the application. Firstly, when user opens the application it directly shows the camera view. User can keep the mobile device on top of the document and move it upwards. Document detection module will do real time document detection and once the whole document is inside the camera view's frame, application will give an audio notification. After the audio notification device will automatically capture the document and sends it to skew detection and correction module. After skew correction captured image will be saved in external storage of the device. The processed image will then send to OCR module which identifies text in image. If no texts identified, application will show camera view again so user can try another document. If OCR module identifies texts in image, those texts are saved as a text file. Finally, texts identified by OCR module will be given as the input to the TTS module, which will read the content in given language. This whole process is illustrated with a use case diagram and a class diagram, see Figure A-1 and Figure A-2 in appendix respectively.

## Chapter 4

### 4. Implementation

This chapter covers design decisions taken when implementing, development environment setup details and technologies used to achieve the final outcome. Also this chapter contains system architecture and key functions.

#### 4.1 Design Decisions

##### 4.1.1 Application for Smartphones

Main objective of this research is to provide an application to visually impaired people so that they can easily read documents in Sinhala. In this era of technology mobile devices plays a significant role in people's lives. People use mobile devices as replacement for many of the traditional equipment. This is mainly because of ease of use and their advance capabilities such as;

- High processing power
- High quality camera
- Inbuilt text to speech engines

Proposed solution in this research needs following hardware and software requirements to implement its functionality.

- Medium to high processing power to run OCR functionality and real-time document detection algorithm
- High quality camera to capture decent input for OCR engine
- Inbuilt text to speech engines so user can listen to content in the captured document

In today's context most of the mobile phones in the market are capable of providing aforementioned hardware and software requirements for an affordable price. And also people are in the process of replacing their personal computers with high performance mobile devices. Hence a mobile application is an ideal choice for the propose solution in this research.

##### 4.1.2 Android Technology

Android is an operating system (OS) for mobiles which is developed by Google, based on a modified Linux kernel and other open source software. This is initially designed for touch screen mobile devices such as tablets and smartphones. Android is the widely used mobile

operating system for mobile devices currently at the market and it has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system, and as of 2017, the Google Play store features over 3.5 million apps. Android's source code is released as an open source project led by Google, but most of Android devices comes to market with combination of open source and proprietary software applications. Since Android is an open source software stack it attracts a massive community of developers and enthusiastic people around the world. This allows developers to run their own application on top of their framework without a hassle [16], [17].

## Android Architecture

Android OS is architected in the form of a software stack comprising applications, an operating system, run-time environment, middleware, services and libraries. This architecture can, perhaps, best be represented visually as outlined in Figure 4-1. Each layer of the stack, and the corresponding elements within each layer, are tightly integrated and carefully tuned to provide the optimal application development and execution environment for mobile devices [18], [19].

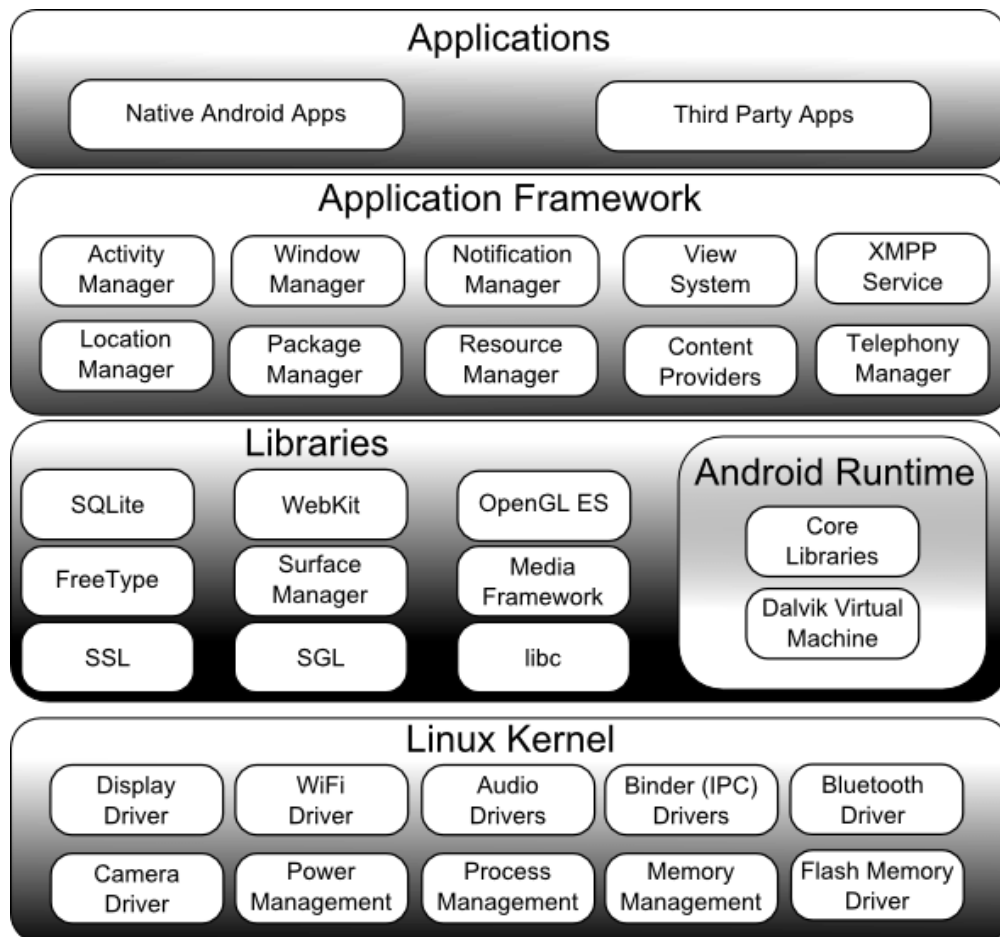


Figure 4-1: Android Architecture

## **4.2 Development Environment**

### **4.2.1 Operating System**

Microsoft Windows is the primary development environment. This is a group of several graphical operating system families which are developed, marketed and sold by Microsoft. Windows OS an advanced and mostly used and popular personal computer OS in the world. With user friendly graphical user interface it attracts a much more user base than other available OS for personal computers.

### **4.2.2 Android Studio IDE**

Android platform's official integrated development environment (IDE) is Android Studio. This is a specially designed software for Android developments which supports any type of Android developments varying from mobile to wearable. This is built on JetBrains' IntelliJ IDEA software [19].

Some of the features this IDE currently holds are instant run where it intelligently identifies recent changes and deliver them without restarting or rebuilding the APK, gradle based build support, rich editor to edit UI elements which allows drag and drops as well, highly accurate emulator and easy support for Git the open source version control system [19].

### **4.2.3 Source Code Repository**

GitLab is the source code management repository used for this research which is lightweight and specially designed version control system for software development with features like issue management, version control, code review and monitoring. GitLab is a web-based Git repository manager using an open source license, developed by GitLab Inc. This offers a free package and some paid packages as well.

## **4.3 Technologies**

### **4.3.1 Android Software Development Kit**

A software development kit (SDK) that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries to build Android applications. Applications are written using the Java programming language and run on Dalvik a custom virtual machine designed for embedded use which runs on top of a Linux kernel.

Since Java language is used for writing applications any developers who are familiar with languages originated from C language can easily be familiar to this development environment. Android Activity is the basic building block which contains an interface. An Android application may contain one or more Activities. Figure 4-2 illustrates life cycle of an Activity [19].

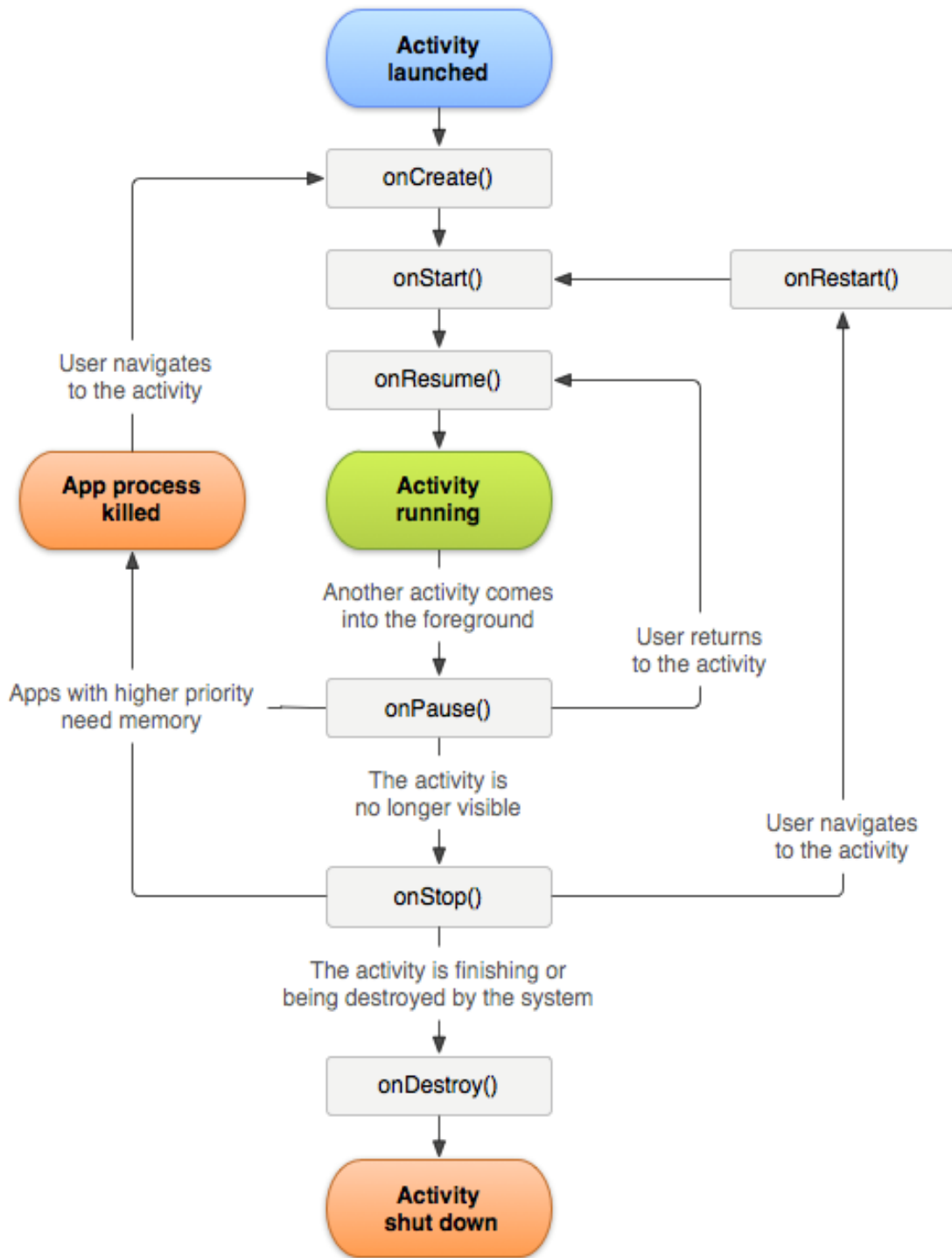


Figure 4-2: Android Activity life cycle

## Android Debug Bridge

Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with a device. Android SDK platform tools includes this adb which facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell which allows developers to run variety of commands on a virtual or real device [19].

### 4.3.2 OpenCV Library

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision which was originally developed by Intel. For Android application this can be integrated as a static library or can install OpenCV Manager from Google Play Store at the initiation stage of the application [20]. Under this research solution the static integration method is used because it is the most user friendly solution for the target audience who are visually impaired. This static integration drastically increased the size of the APK therefore OpenCV support were only added for x86 and armeabi-v7a CPU architecture variations.

For static integration firstly we have to download the latest OpenCV SDK and import to project as an independent module. Then OpenCV module is added as a dependency module. Then adding required architecture files to jniLibs directory and rebuild the project.

### 4.3.3 Tesseract OCR

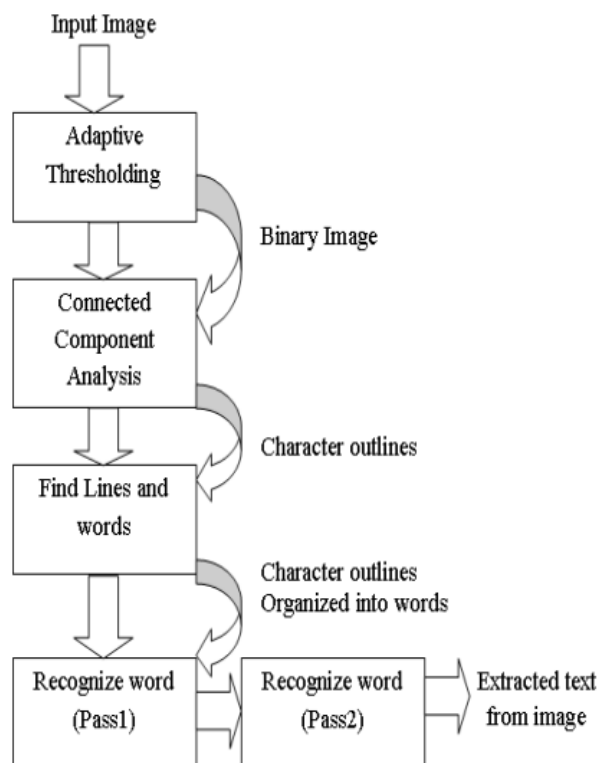


Figure 4-3: Architecture of Tesseract OCR engine

Tesseract is the most popular and widely used OCR engine which is an open source library. Figure 4-3 shows the architecture of Tesseract OCR engine [23]. It is well known for highly accuracy and its ability to recognize characters in over 100 languages from a wide variety of image formats. Tesseract works on Windows, Linux and Mac OSX. Also it can work on Android and iOS as well though these are still under continuous development. This engine can be easily trained for languages to recognize text and when it comes to Android it has a wrapper which can be integrated and is available as open source project in GitHub.

#### 4.3.4 Android TTS

This application integrates the default Text to Speech capability also known as “speech synthesis”, which is included in Android platform. TTS enables an Android device to "speak" text in various languages. Although all Android-powered devices that support the TTS functionality ship with the TTS-engine some devices have limited storage and may lack the language specific resource files. This application integrates Google’s TTS engine. If device has Google TTS engine, Sinhala TTS functionality can also be used but language pack selection should be done manually currently.

#### 4.4 System Architecture

This solution is an Android based solution which runs on top of Android SDK. Following figure illustrate the architecture diagram of the application.

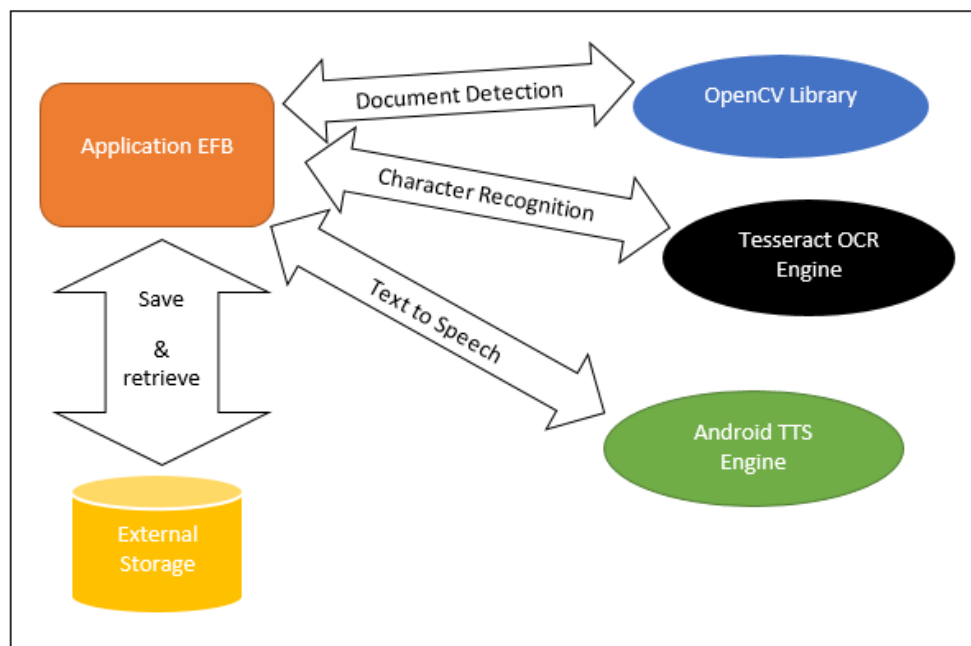


Figure 4-4: Architecture Diagram

## 4.5 Execution Flow of Application

Once user opens application for the first time if user uses a device with an OS Android 6.0 or higher first user will have to give necessary permissions if not application will close down. When necessary permissions are acquired application direct to a camera view and provides an audio guidance asking user to keep the mobile above the document and move the mobile phone upwards. When whole document is within the camera view it will give a specific notification sound then user has to hold the device steadily until application automatically captures document with camera shutter sound notifying the image is captured. Then image will get through OCR engine for character recognition process and when it is over application will automatically start reading the document from beginning till end without stopping. Once TTS has finished reading out text it will give a specific sound and will return back to camera view, so user can either proceed forward with another document or can exit the application.

## 4.6 Key Functions

### 4.6.1 Document Detection

Image processing is done with OpenCv for Android to detect edges and this application detects document by identifying its edges. This task is performed real-time with following steps.

- Blurring the image so that edges are enhanced and can detect easily. Median blur function is used.
- Mixing gray image and blurred image with color image and passes through edge detection algorithm.
- Retrieve the largest rectangular in the image and draws a green line indicating the document identified.
- Skew correcting the image before passing it to OCR engine.

This function requires a lot of CPU processing power and memory, therefore while implementing this lot of hours were taken to implement the system avoiding memory leaks and system crashes. Following are some of the important code segments in the code base.

```
static double angle(Point pt1, Point pt2, Point pt0) {  
    double dx1 = pt1.x - pt0.x;  
    double dy1 = pt1.y - pt0.y;  
    double dx2 = pt2.x - pt0.x;  
    double dy2 = pt2.y - pt0.y;  
    return (dx1 * dx2 + dy1 * dy2) / Math.sqrt((dx1 * dx1 + dy1 * dy1) * (dx2 * dx2 + dy2 * dy2) + 1e-10);  
}
```

Figure 4-5: Skew correction algorithm



Figure 4-5 illustrates the skew correction algorithm which is for to correct angled documents. Figure 4-6 in below is the border detection algorithm. Firstly, ‘Median Blur’ filtering technique is used to enhance edges then ‘Canny Edge Detection’ algorithm is used to identify the rectangular contours in the image. ‘Dilation’ operation removes holes between potential edge segments and make them sharp. Finally identified contours are stored in a list and checks for the largest contour.

```

for (int c = 0; c < 3; c++) {
    int ch[] = {c, 0};
    Core.mixChannels(blurredChannel, gray0Channel, new MatOfInt(ch));
    int thresholdLevel = 1;
    for (int t = 0; t < thresholdLevel; t++) {
        if (t == 0) {
            Imgproc.Canny(gray0, gray, threshold1: 10,
                threshold2: 20, apertureSize: 3, L2gradient: true);
            Imgproc.dilate(gray, gray, new Mat(), new Point( x: -1, y: -1), iterations: 1);
        } else {
            Imgproc.adaptiveThreshold(gray0, gray, thresholdLevel,
                Imgproc.ADAPTIVE_THRESH_GAUSSIAN_C,
                Imgproc.THRESH_BINARY, blockSize: (src.width() + src.height()) / 200, t);
        }
    }
    Imgproc.findContours(gray, contours, new Mat(), Imgproc.RETR_LIST,
        Imgproc.CHAIN_APPROX_SIMPLE);

    for (MatOfPoint contour : contours) {
        double area = Imgproc.contourArea(contour);
        if (area > maxArea) {
            maxArea = area;
            approxCurve = new MatOfPoint2f();
            MatOfPoint2f temp = new MatOfPoint2f(contour.toArray());
            Imgproc.approxPolyDP(temp, approxCurve,
                epsilon: Imgproc.arcLength(temp, closed: true) * 0.02, closed: true);
            if (approxCurve.total() == 4 && area >= maxArea) {
                double maxCosine = 0;
                List<Point> curves = approxCurve.toList();
                for (int j = 2; j < 5; j++)
                {
                    double cosine = Math.abs(angle(curves.get(j % 4),
                        curves.get(j - 2), curves.get(j - 1)));
                    maxCosine = Math.max(maxCosine, cosine);
                }
                if (maxCosine < 0.3) {
                    maxArea = area;
                    maxId = contours.indexOf(contour);
                }
            }
        }
    }
}

```

Figure 4-6: Border detection algorithm

When the system detects a document, it gives user an audio notification asking user to hold the device steadily. System automatically captures the document in five seconds after giving that audio notification and sends it to the OCR engine. Android requires to obtain necessary permissions in order to write the image into internal storage of the device (android.permission.WRITE\_EXTERNAL\_STORAGE) and camera permissions to use the camera (android.permission.CAMERA) inside the application. After Android 6.0 (API level 23) if app's target SDK is 23 or higher permissions should be declare in app's manifest file and also needs to acquire at the runtime.

## 4.6.2 Character Recognition

The image of the document which was captured with above function is then passed to Tesseract OCR for character recognition. At the initialization OCR checks if dataset is within the external memory of the device if not it copies the dataset file from internal memory of the application to external memory of the device. Figure 4-7 illustrates the code segment for that. Since a language toggle is introduced OCR needs to be initialized whenever language is changed with the relevant dataset file.

```
public TessOCR(Context context) {
    // TODO Auto-generated constructor stub      this.context = context;
    datapath = Environment.getExternalStorageDirectory().getAbsolutePath() + "/efb/";
    File dir = new File( pathname: datapath + "/tessdata/");
    File file = new File( pathname: datapath + "/tessdata/" + KeyValues.language+".traineddata");
    if (!file.exists()) {
        Log.d( tag: "mylog", msg: "in file doesn't exist");
        dir.mkdirs();
        copyFile(context);
    }else {
        mTess = new TessBaseAPI();
        Log.e( tag: "language", KeyValues.language);
        mTess.init(datapath, KeyValues.language); //Auto only
        mTess.setPageSegMode(TessBaseAPI.PageSegMode.PSM_AUTO_ONLY);
    }
}
```

Figure 4-7: Initializing OCR engine

### 4.6.3 Text to Speech

Text identified by OCR engine is taken as a string output and then passes to the TTS which is initialized at the run time. Then TTS will read the text continuously until whole text input is finished.

```
private void sayText(final String string){
    textToSpeech=new TextToSpeech( context: this, (status) → {
        if (status == TextToSpeech.SUCCESS) {
            int result = textToSpeech.setLanguage(Locale.US);
            if (result == TextToSpeech.LANG_MISSING_DATA
                || result == TextToSpeech.LANG_NOT_SUPPORTED) {
                Log.e( tag: "TTS", msg: "This Language is not supported");
            }else {
                mTessOCR = new TessOCR( context: OCRActivity.this);
                textToSpeech.speak(string, TextToSpeech.QUEUE_FLUSH, params: null);
            }
        } else {
            Log.e( tag: "TTS", msg: "Initilization Failed!");
        }
    });
}
```

Figure 4-8: Initializing TTS

## Chapter 5

### 5. Evaluation and Results

This chapter covers evaluation methods undertaken to evaluate the proposed solution and discusses about test results of each section.

#### 5.1 Functional Evaluation

Functional evaluation was carried out on main three sub systems of the proposed solution. Following are some of the testing devices used for this evaluation mechanism.

Device	Android OS Version	Is compatible with EFB?
Samsung Galaxy J7	7.0	Yes.
Samsung Galaxy Tab S	6.0	Yes.
OnePlus 3	8.0	Yes.
HTC 10	8.0	Yes.
HTC Desire Eye	7.0	Yes.

Table 5-1: Tested devices

When auto focus functionality is present in device the input for OCR was good and accuracy of the system was high. Therefore it is highly recommended to use devices with auto focus functionality. All three Samsung devices used Samsung TTS and even for Sinhala the reading accuracy and clearance of audio output was in good quality. Other devices used Google TTS with English/Sinhala language and audio output was comparatively low than Samsung default TTS.

##### 5.1.1 Document Detection Accuracy

Document detection and capturing it automatically, is a core function in this system as this application targets a visually impaired user base. For this purpose OpenCV, an open source library was integrated. This identifies documents by detecting square/ rectangles in the camera view by identifying 90 degree angles between the borders of the document. This algorithm was written in a way that it detects the largest rectangular contour within the given view, therefore the size of the document doesn't have to be in the A4 size. Figure 5-1 illustrate how document

detection happens in real time. When it is identifying document it highlights document's edge in green color.

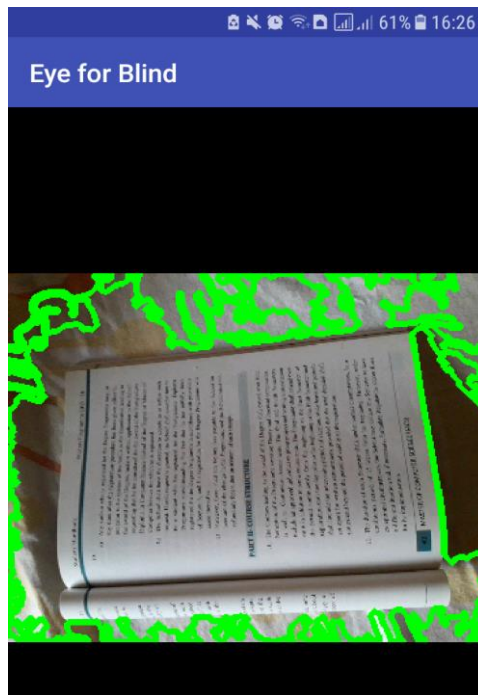


Figure 5-1: Real time document detection

In Figure 5-1 the application tries to detect document which is not a simple task as background cannot be clearly distinguish, therefore it is recommended to use a good contrast background to detect the document more accurately as shown below.

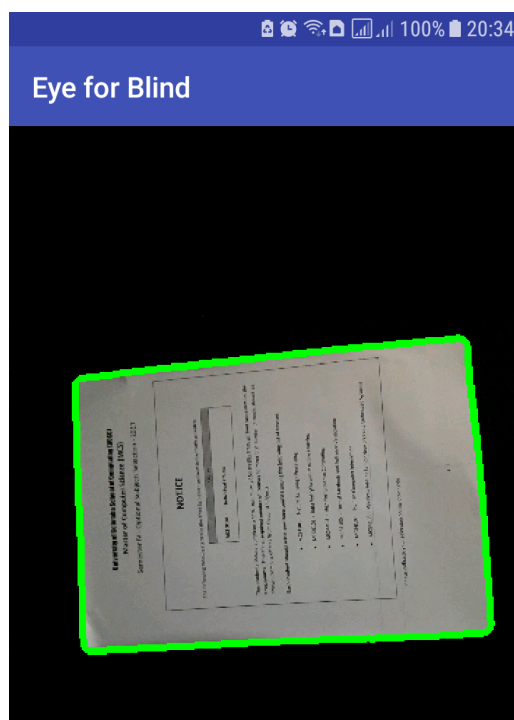


Figure 5-2: High accuracy in document detection

In Figure 5-3 shows how document detection is dependable on the background and foreground color contrast.

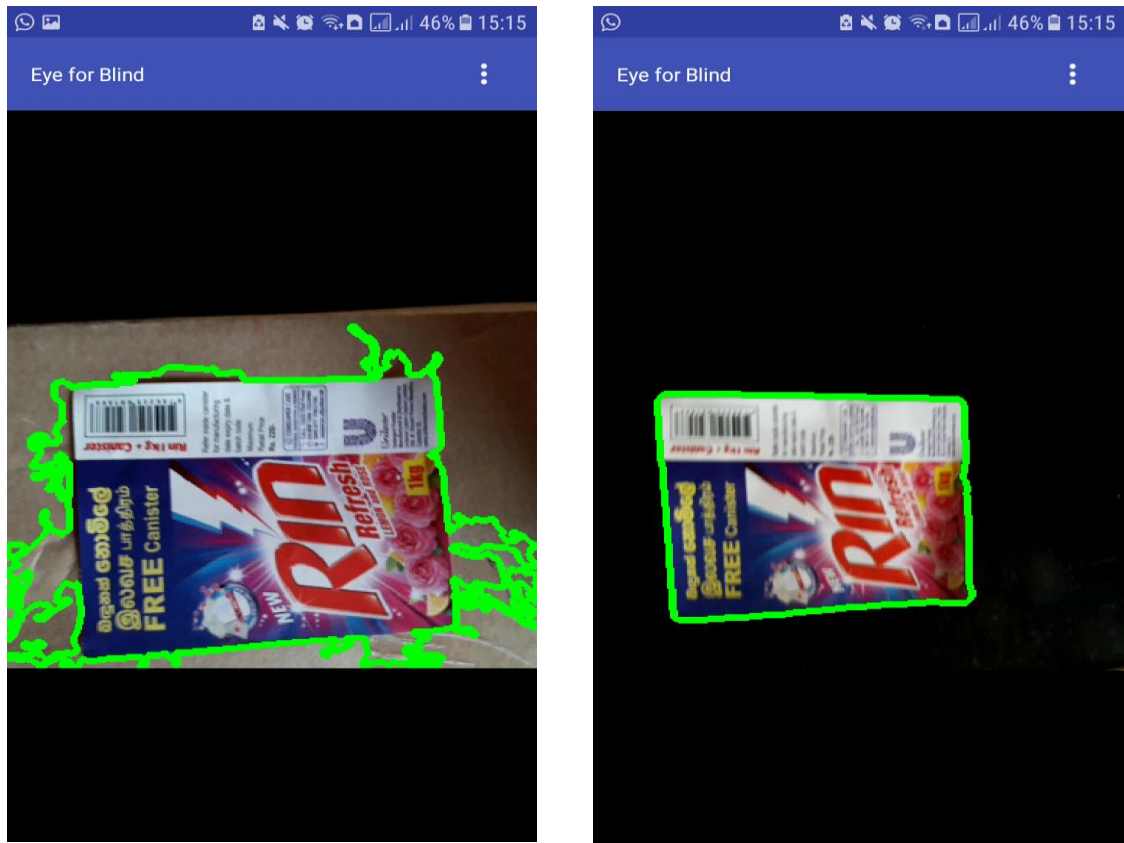


Figure 5-3: Importance of background color contrast in document detection

Different document sizes were tested other than A4 size which includes newspaper articles, labels of household items and text books with variety page sizes. And system was able to detect contours accurately in every document tested regardless of the documents size provided that background and foreground color contrast is high. Following table shows the summarized test results.

Document	Width (cm) x Height (cm)	Can EFB detect it?
Newspaper article	13 x 18	Yes.
Label of a household item	8.8 x 10.8	Yes.
A4 document	21 x 29.7	Yes.
Novel	13 x 20	Yes.
Text book	18 x 20	Yes.
Leaflet	15 x 15	Yes.

Table 5-2: Tested Documents

## 5.1.2 Optical Character Recognition Accuracy

Detected document is captured and passed in to the Tesseract OCR engine. Though this is proposed only for Sinhala language I have added English OCR functionality as well. Multiple languages can be integrated in to OCR engine by giving the relevant language dataset file. When user toggles between languages it gives an audio notification stating which language is selected. OCR accuracy totally depends on how well the trained dataset that is used in Tesseract engine. Also input image quality affects the OCR's accuracy. As shown below the accuracy of English language is impressively higher than accuracy of Sinhala language.



Figure 5-4: Comparison of Sinhala and English OCR function

Following table, shows some documents and their corresponding OCR engine's text output. Table 5-3 and 5-4 shows English and Sinhala OCR outputs respectively.

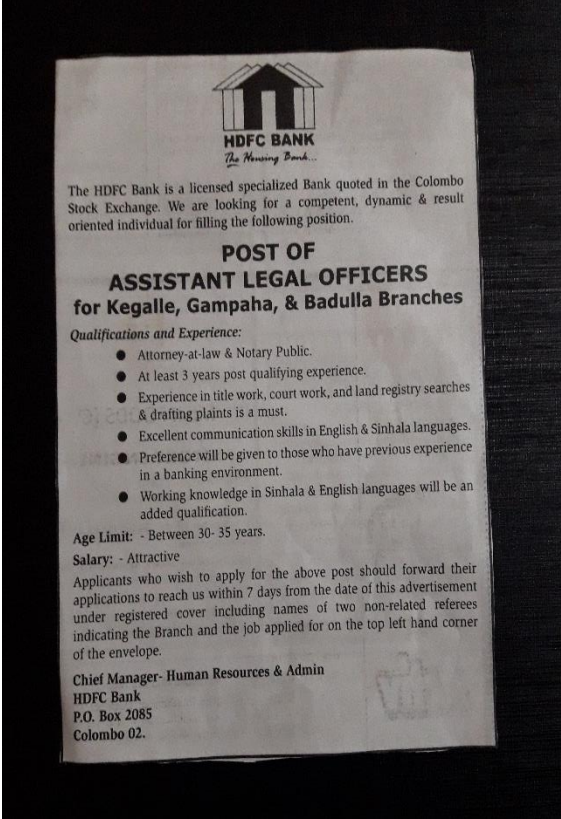
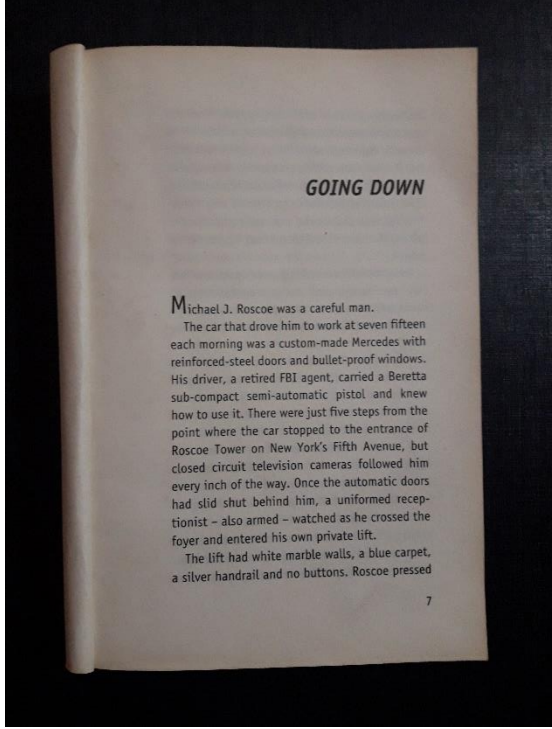
Document	Text Output
	<p>Fibre BANK 75. Mw, Bad...</p> <p>The 11D1-'C Bank is a licensed specialized Bank quoted in the Colombo Stock Exchange. We are looking for a competent, dynamic &amp; result oriented individual for filling the following position.</p> <p>POST OF</p> <p>ASSISTANT LEGAL OFFICERS for Kegalle, Gampaha, &amp; Badulla Branches</p> <p>Qualifications and Experience: Attorney-at-law &amp; Notary Public. At least 3 years post qualifying experience. Experience in title work, court work, and land registry searches &amp; drafting plaints is a must. Excellent communication skills in English &amp; Sinhala languages. Preference will be given to those who have previous experience in a banking environment. Working knowledge in Sinhala &amp; English languages will be an added qualification.</p> <p>Age Limit: - Between 30- 35 years.</p> <p>Salary: - Attractive Applicants who wish to apply for the above post should forward their within 7 days from the date of this advertisement</p> <p>including names of two non-related referees the job applied for on the top left hand corner</p> <p>applications to reach us under registered cover indicating the Branch and</p> <p>of the envelope. Chief Manager- Human Resources &amp; Admin</p> <p>HDFC Bank PO. Box 2085 Colombo 02.</p>
	<p>GOING DOWN</p> <p>Michael J Roscoe was a careful man</p> <p>The car that drove him to work at seven fifteen each morning was a custom-made Mercedes with reinforced-steel doors and bullet-proof windows. His driver, a retired FBI agent, carried a Beretta sub-compact semi-automatic pistol and knew how to use it. There were just five steps from the point where the car stopped to the entrance of Roscoe Tower on New York's Fifth Avenue, but closed circuit television cameras followed him every inch of the way. Once the automatic doors had slid shut behind him, a uniformed receptionist - also armed - watched as he crossed the</p> <p>foyer and entered his own private lift. The lift had white marble walls, a blue carpet, a silver handrail and no buttons. Roscoe pressed</p> <p>7</p>

Table 5-3: English OCR output





devices get Google TTS as default TTS and Samsung devices' have their own TTS engine and it does a pretty much good job with clear audio output.

Sinhala language pack is available on Google TTS with a female voice file. This also works well if the OCR output is accurate. This Sinhala language file even clearly reads out a dialogue with more similar to a human dialogue.

## 5.2 Non Functional Evaluation

For this application nonfunctional requirements are also important as this application is designed for visually impaired people. More importance was given to user interface design, user interface was kept really simple and instead of visual notification mechanisms like alerts or toasts, audio notifications and guidelines are given.

### 5.2.1 Usability Evaluation

To test how user friendly the application and how easy it is to use for a visually impaired person is evaluated using test subjects. This group had 25 people whose age are in the range of 20 years to 40 years and they all are well experienced with Android based smartphones. These people weren't visually impaired, so they were asked to use this application while keeping their eyes closed. They were given similar devices which have accessibility functionalities so they could open this application through voice commands. Before the test a brief introduction was given about the purpose of the application and some guidance like how to use accessibility functions to open the application. A printed single page English and Sinhala documents were also given. After the test they were given a questionnaire to obtain feedback. Figure 5-4 shows the template of the questionnaire.

Feedback	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
1. Ease of use in first time					
2. Audio guidance and notifications are helpful					
3. Language toggle option is useful					
4. Text to speech output is clear					
5. Overall efficiency of the application is adequate					

Figure 5-5: Questionnaire format

Collected feedback from the aforementioned individuals can be summarized as shown in table 5-5 below.

<b>Feedback</b>	<b>Strongly agree</b>	<b>Agree</b>	<b>Neutral</b>	<b>Disagree</b>	<b>Strongly disagree</b>
1. Ease of use for first time	4	20	1	-	-
2. Audio guidance and notifications are helpful	23	2	-	-	-
3. Language toggle option is useful	18	5	2	-	-
4. Text to speech output is clear	15	4	5	1	-
5. Overall efficiency of the application is adequate	13	8	4	-	-

Table 5-5: Test users' feedback summary

According to the figures in Table 5-3, user friendliness of the application is at an acceptable level as the negative feedback is negligible compared to positive feedback received. Implementing audio based notification and guidance has added value to user experience as many people have strongly agreed to it. Text to speech system surely needs an improvement as it has received most less positive compared to others. But with overall efficiency feedbacks not being in negative region we can conclude that usability of this application is adequate.

## **Chapter 6**

### **6. Conclusion and Future Work**

This research thesis describes the contribution done to implement a mobile based Sinhala document reader to support visually impaired Sri Lankans to fulfil their craving for information and knowledge gaining. With this application the visually impaired people will be able to capture a document with the camera on their mobile device and listen to the document's content comfortably with lesser effort. This section covers the research conclusion and future work.

#### **6.1 Research Conclusion**

In overall this research solution was promising and was able to address the core aspects of the research and has potential improvements. This application uses audio notifications to guide users so that they can use the application more effectively. Implementation of this application in a more user friendly manner was challenging yet was the most value adding function to this application. With border detection this solution detects a document in front of the camera, for this purpose OpenCV static integration was used. Though static integration is not the recommended approach for Android it was essential to minimize the user engagement to enhance convenience for the end user. Identified document is automatically captured by application and sends it to OCR engine. Tesseract is the OCR engine for the application and integrating this was not difficult as it was already available as an open source library which can integrate in to custom applications. Language toggle provides to switch between Sinhala and English OCR functionalities but a common TTS support is used for both the languages. Once a document is read out user can reuse the application without restarting the application this is really helpful if user wants to read a multi paged document.

#### **6.2 Future Work**

This system was developed to demonstrate that an Android based mobile solution can be developed to fulfill this need of visually impaired people in Sri Lanka. However though this system shows the effectiveness of the proposed solution it needs to be improved if this is to present as a commercially available application in Google Play Store. To improve the accuracy of system as a whole firstly the accuracy of OCR should be addressed as it is the input for TTS. OCR engine's accuracy can be improved with following potential mechanisms. By enhancing

the input image and by removing noise in input image the OCR engine can recognize characters very efficiently. Also a well-trained dataset can be generated locally and can replace the trained dataset file used for OCR engine to increase character recognition accuracy. A dictionary lookup function can also be integrated as a part of OCR process to minimize the errors in optical character recognition process.

Text to Speech engine currently uses the device's default TTS engine. Although Google TTS engine provides the Sinhala language voice pack it cannot be directly accessed via programming. Therefore it is better to implement Sinhala voice dataset for a TTS engine which can be integrated into the application because it will be more user-friendly for the target audience if user doesn't have to configure application on his/ her end.

This application can be developed for multiple language support. This currently offers English and Sinhala language support. Potential language integration is Tamil, so that this application can cater all visually impaired community in Sri Lanka.

## References

- [1] Visual Impairment and Blindness - World Health Organization. [Online]. Available: <http://www.who.int/mediacentre/factsheets/fs282/en>. [Accessed: 20-Jun-2017].
- [2] Sandeep Kaur and Rekha Bhatia. A Systematic Review of Optical Character Recognition Techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, 6, September 2016.
- [3] Najib Ali Mohamed Isheawy and Habibul Hasan. Optical Character Recognition (OCR) System. *IOSR Journal of Computer Engineering*, 17(2), March 2015.
- [4] Tesseract (software). [Online]. Available: [https://en.wikipedia.org/wiki/Tesseract\\_software](https://en.wikipedia.org/wiki/Tesseract_software) [Accessed: 20-Jun-2017].
- [5] Optical character recognition. [Online]. Available: [https://en.wikipedia.org/wiki/Optical\\_character\\_recognition](https://en.wikipedia.org/wiki/Optical_character_recognition). [Accessed: 20-Jun-2017].
- [6] What is OCR?. [Online]. Available: [https://abbyy.technology/en:kb:tip:what\\_is\\_ocr](https://abbyy.technology/en:kb:tip:what_is_ocr). [Accessed: 05-Aug-2017].
- [7] Text to Speech Overview. [Online]. Available: <http://www.voicerss.org/tts/>. [Accessed: 09-Aug-2017].
- [8] A.J. Clark, Korin Richmond and Simon King. Festival 2 – build your own general purpose unit selection speech synthesizer. In Proc. *5th ISCA workshop on speech synthesis*. 2004.
- [9] R. Weerasinghe, A. Wasala, V. Welgama and K. Gamage, "Festival-si: A Sinhala Text-to-Speech System", in *10th Conference on Text, Speech and Dialogue (TSD 2007)*, Pilsen, Czech Republic, 2007.
- [10] Michal Kaszczuk and Lukasz Osowski. Evaluating Ivona Speech Synthesis System for Blizzard Challenge. [Online]. Available: <http://www.ivosoftware.com>
- [11] eSpeak. [Online]. Available: <https://en.wikipedia.org/wiki/ESpeakNG>. [Accessed: 20-Jun-2017].

- [12] W.M.C. Bandara, S.V. Bulathsinghala, W.M.S. Lakmal and T.D. Liyanagama, ‘AMORA: Sinhala Text to Speech System’, Undergraduate, Department of Computer Science and Engineering, University of Moratuwa, 2009.
- [13] P.G.D.N. Pushpakumara, ‘Android Cam Translator Application’, Postgraduate, University of Colombo School of Computing, 2013.
- [14] A.R.L. Madhusanka, ‘Tamil to Sinhala Augmented Reality Translation’, Postgraduate, University of Colombo School of Computing, 2016.
- [15] Google Play [Online]. Available: <https://play.google.com>. [Accessed: 11-June-2017].
- [16] Android (Operating System). [Online]. Available: [https://en.wikipedia.org/wiki/Android\\_\(operating\\_system\)](https://en.wikipedia.org/wiki/Android_(operating_system)). [Accessed: 03-Feb-2018].
- [17] About the Android Open Source Project. [Online]. Available: <https://source.android.com>. [Accessed: 03-Feb-2018].
- [18] An Overview of the Android Architecture. [Online]. Available: <http://www.techotopia.com>. [Accessed: 04-Feb-2018].
- [19] Android Developers. [Online]. Available: <http://developer.android.com>. [Accessed: 05-Feb-2018].
- [20] Android – OpenCV Library. [Online]. Available: <https://opencv.org/platforms/android>. [Accessed: 05-Aug-2017].
- [21] S. Ronanki, S. Reddy, B. Bollepalli, and S. King. DNN-based Speech Synthesis for Indian Languages from ASCII text. In Proc. *9th ISCA Speech Synthesis Workshop (SSW9)*, Sunnyvale, CA, USA. September 2016.
- [22] Basic Android Accessibility – making sure everyone can use what you create. [Online]. Available: <https://codelabs.developers.google.com/codelabs/basic-android-accessibility/index.html>. [Accessed: 05-Feb-2018].

- [23] S. Patel, A. Patel, and D. Patel. Optical Character Recognition by Open Source OCR Tool Tesseract. *International Journal of Computer Applications* (0975 – 8887). October 2012.
- [24] G.Vamvakas, B.Gatos, N. Stamatopoulos, and S.J.Perantonis. A Complete Optical Character Recognition Methodology for Historical Documents. *The Eighth IAPR Workshop on Document Analysis Systems*. 2008.
- [25] Fork of Tesseract Tools for Android. [Online]. Available: <https://github.com/rmtheis/tess-two>. [Accessed: 15-July-2017].
- [26] Optical Character Recognition – Edubilla. [Online]. Available: <http://www.edubilla.com/invention/optical-character-recognition>. [Accessed: 15-Jan-2018].
- [27] Muhammad Tahir Qadri and Muhammad Asif. Automatic number plate recognition system for vehicle identification using optical character recognition. In *Education Technology and Computer (ICETC'09) International Conference on IEEE*. 2009.



# Appendix A

## Application Design Diagrams

### A.1 Use Case Diagram of the Application

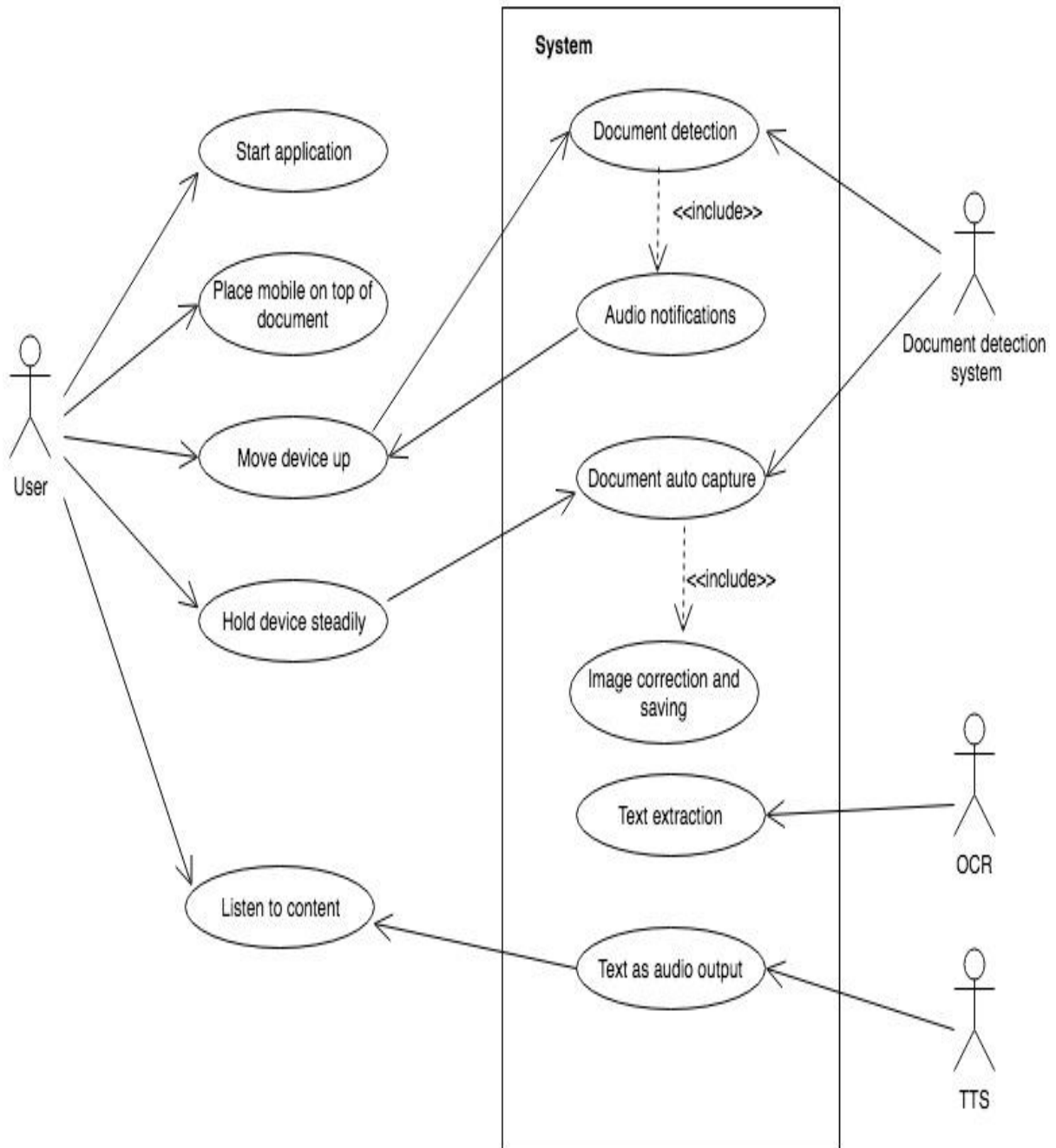


Figure A-1: Use case diagram of the system

## A.2 Sequence Diagram of the Application

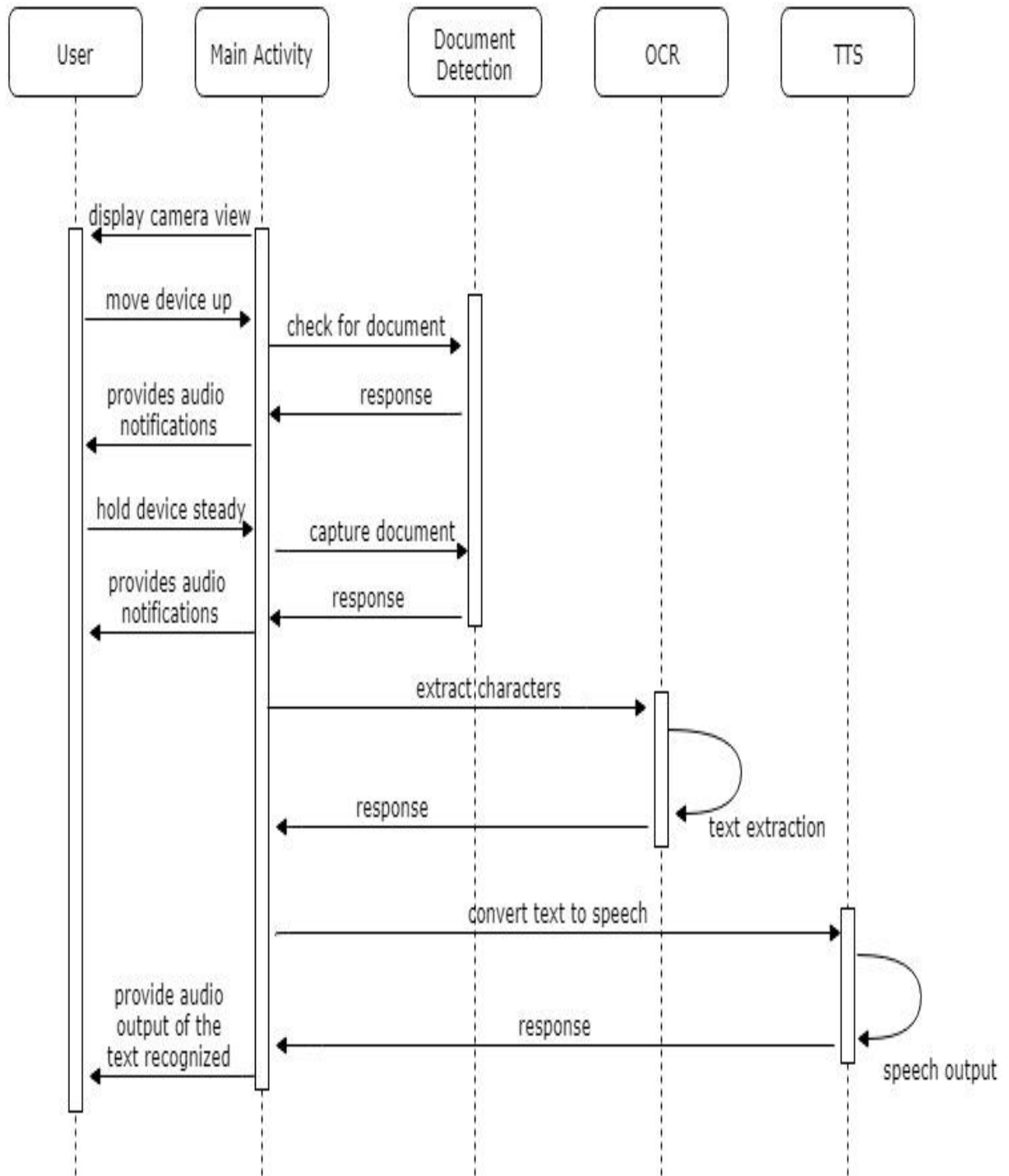


Figure A-2: Sequence diagram for the system