A Project Report

on

TAP TAP SEARCH

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

by

B.Alekhya (16WH1A1205)

A.Sai Sanjana(16WH1A1209)

K.Sanjana(16WH1A1231)

Under the esteemed guidance of

Ms.S. Rama Devi
Associate Professor

Department of Information Technology

BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

(Accredited by NBA & NAAC '**A**' Grade, Approved by AICTE, and
Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

April 2020

DECLARATION

We hereby declare that the work presented in this project entitled **"TAP TAP SEARCH"**submitted towards completion of the major project in IV year of B.Tech IT at "BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN**"**, Hyderabad is an authentic record of our original work carried out under the esteem guidance of Ms.S.Rama Devi, Associate Professor, IT department.

B.Alekhya
(16WH1A1205)

A.Sai Sanjana
(16WH1A1209)

K.Sanjana
(16WH1A1231)

BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

(Accredited by NBA & NAAC 'A' Grade, Approved by AICTE, and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

Department of Information Technology

Certificate

This is to certify that the Project report on "**TAP TAP SEARCH**" is a bonafide work carried out by **B.Alekhya(16WH1A1205) , A.Sai Sanjana(16WH1A1209) , K.Sanjana(16WH1A1231)** in the partial fulfillment for the award of B.Tech degree in **Information Technology, BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide	**Head of the Department**
Ms. S. Rama Devi	**Dr. ArunaRao S L**
Associate Professor	**Professor and HOD**
Department of IT	**Department of IT**

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K. V. N. Sunitha, Principal**, BVRIT Hyderabad, for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr**. **ArunaRao S L, Head**, Dept. of IT, BVRIT Hyderabad for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Ms. S. Rama Devi, Associate Professor, Department of IT,** BVRIT Hyderabad for her constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project coordinator, all the faculty and staff of IT Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

B. Alekhya
(16WH1A1205)
A. Sai Sanjana
(16WH1A1209)
K. Sanjana
(16WH1A1231)

Abstract

A Tap Tap Search is an Android Application. The main idea of this project is to recognize the text character and convert it into a speech signal. Here, we developed a useful text-to-speech synthesizer in the form of a simple application that converts inputted text into synthesized speech and reads out to the user. We also implemented Speech to Text where it takes the speech from the user and it converts into text and also implemented notification reader where if the user receives any text message from any one then it will tell what is the message received and the sender's number. In this application we implemented a feature of uploading text files and the android takes the text and that will be converted into speech. The development of a text to speech synthesizer will be of great help to people with visual impairment and make making through large volumes of text easier.

LIST OF FIGURES

LIST OF ABBREVATIONS

Term/Abbreviation	Definition
TTS	Text to Speech
OCR	Optical Character Recognition
NLP	Natural Language Processing
DSP	Digital Signal Processing
STTR	Speech to Text Repoter

TABLE OF CONTENTS

1. INTRODUCTION

Text-to-speech synthesis -TTS - is the automatic conversion of a text into speech that resembles, as closely as possible, a native speaker of the language reading that text. Text-to speech synthesizer (TTS) is the technology which lets computers speak to you. The TTS system gets the text as the input and then a computer algorithm which is called TTS engine analyses the text, pre-processes the text and synthesizes the speech with some mathematical models. The TTS engine usually generates sound data in an audio format as the output.

The Text-to-Speech (TTS) synthesis procedure consists of two main phases. The first is text analysis, where the input text is transcribed into a phonetic or some other linguistic representation, and the second one is the generation of speech waveforms, where the output is produced from this phonetic and prosodic information. These two phases are usually called high and low-level synthesis. The input text might be for example data from a word processor, standard ASCII from email, a mobile text-message, or scanned text from a newspaper. The character string is then pre-processed and analyzed into phonetic representation which is usually a string of phonemes with some additional information for correct intonation, duration, and stress. Speech sound is finally generated with the low-level synthesizer by the information from high-level one. The artificial production of speech-like sounds has a long history, with documented mechanical attempts dating to the eighteenth century.

1.1 Objective

Our main objective is to create an android application which converts text to speech and efficiently works for disabled people.

1.2 Existing System

Manually typing a text, viewing notifications and any messages will be difficult when the user is busy. It is also time consuming as the user always has to check their mobile. Also there are applications that will convert text to voice but it will be active only when the user receives the message.

1.3 Proposed System

The Text-to-voice conversion application will always run in the background. Along with converting text to voice, this application also takes voice and work accordingly. Unlike other apps that will read only the message got from the recipient only when it is send, this app will read out all notifications and messages inside each notification and message inside each notification as per the user convenience. Whenever the user asks, it will respond and read out those messages which they want.

2. LITERATURE SURVEY

We have done some literature survey where we have gone through some similar mobile applications for visually impaired people. According to those applications we have made our application like if they were not included any features and all. Our application is done in such a way that we have introduced new additions in which other applications are not yet included. Below are some of the applications we have gone through.

- **Scan Life Barcode and QR Reader**: Visually impaired people can also benefit from a miraculous application called Scan Life Barcode and QR Reader integrated in the technology. The user only needs to click picture and the application does the rest of the magic. The app can read QR and UPC codes. After the codes are scanned, the application starts reading the embedded string as QR codes.
- **Magnify**: This application helps visually impaired individuals read tiny prints. To get the best results, keep the phone at least four inches away from the material to be read. You can also turn the flashlight on or off manually to save battery. The app also supports features such as double tap to zoom in and out, long press to turn lights on and off, and single tap to maintain focus.
- **Talk Back**: This application is a part of Google's Android Accessibility Service. It is developed to assist visually impaired people to use their cell phones easily. The application also reads out the texts loud, while the movements of the user are carefully evaluated and spoken by the app. In order to enable this app, go to Settings, Accessibility, and enable Talk Back service.
- **Messages Keyboard:** With this application, users can type messages quickly by using one finger or one hand only. There is no problem such as AutoCorrect. The key board has large size fonts so users can see them easily. You can change the colors, use Swype gesture, search through dictionaries, change keyboard layout, haptic feedback, shape, size, and many other features of the application.

- **Ultra Magnifier:** This application helps users get better vision of anything on your phone. To get best results, put the device at least 10 cm away from the object. This application works best with cell phones that do not have locked Camera API's. Users can also change scenes of objects and also select form sepia, solarized effects, negative color, aqua and mono Ciba Vision Air Optix Aqua.
- **Walky Talky:** This application in the Android phones is developed for visually impaired and blind people to walk easily. The application has a navigation aid to help people walk through the streets. Users receive instant updates on current location as the application includes a compass to point directions. If users go off track, the application will vibrate automatically to indicate about it.
- **Ideal Accessibility Installer:** This application is also called Platform Access Installer. It contains different packages such as KickBack, TalkBack, and SoundBack (TKS) applications for visually impaired and blind people. These applications add audible, vibration and spoken feedback to Android devices. After you have installed the application, go to Settings, Accessibility and check the applications that you want.
- **NoLED:** The application displays custom notification dots or icons over the screen so the user can know immediately that there's a notification. Users can receive notifications regarding voice messages, text messages, GoogleTalk notifications, missed calls, calendar events, emails, charging activities and third party applications Ciba Vision Air Optix Aqua.
- **Classic Text To Speech Engine:** The SVOX mobile Voices has developed an application called Classic Text to Speech Engine to help visually impaired people. This application is an integration of more than forty female and male voices to read out users' texts, translations, e-books and even navigation. It also reads PDF documents, e-books, as well as helps with pronunciation.

- **Classic Text to Speech Engine:** The Classic Text to Speech Engine app is a combination of over 40 male and female voices that supports people by reading out aloud their texts, e-books, translations, and even navigation. The app features voice support in key areas like navigation, as it will keep you guided when you are driving. It also reads aloud your favorite e-book or PDF documents, making it an eye-free application. Not only that, it also helps users with their pronunciation. This classic application installs a TTS engine, making it compatible with other applications, as well.
- **Be My Eyes:** "Lend your eyes to the blind" is their tagline and Be My Eyes is very similar to Be Specular in that the app matches a visually impaired user with a sighted volunteer. The difference is that you are connected through a live video connection and the sighted volunteer can tell the user what they see when the user points their phone at something using the rear-facing camera. Watch our Be My Eyes video to see it in action.
- **Color ID:** Color Identifier uses the camera on your phone or tablet to identify and speak the names of colours in real-time. It's an augmented reality app that helps you discover the colours around you. There are many free apps that do this though so have a play and if you don't like it try one of the others, such as, Aid Colors, Color Visor and Aipoly Vision.
- **Cam Find:** Cam Find allows you to do an online search by simply taking a photo of an object – the app then uses mobile visual search technology to tell you what it is. It provides fast results with no typing necessary. Snap a picture and learn more. Search results include related images, local shopping results and a selection of web results, which are also easy to share through the app.

3. FUNCTIONAL REQUIREMENTS

3.1 Software Requirements

- Operating System : Windows10 (64 bit)
- Ide : Android Studio 3
- Technologies : Java
- Other Technologies : Optical Character Recognition(OCR),Text To Speech(TTS)

3.2 Hardware Requirements

- Hardware : Core i5
- Speed : 1.70GHz
- RAM : 8GB
- Hard Disk : 500 GB

3.1.1 Java

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. It is intended to let application developers write once, run anywhere, meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but it has fewer low-level facilities than either of them.

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java platform. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun had relicensed most of its Java technologies under the GNU General Public License.

Meanwhile, others have developed alternative implementations of these Sun technologies,

such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).

3.1.1.1 Principles

There were five primary goals in the creation of the Java language:

- It must be simple, object-oriented, and familiar.
- It must be robust and secure.
- It must be architecture-neutral and portable.
- It must execute with high performance.
- It must be interpreted, threaded, and dynamic.

3.1.1.2 Special Classes

- **Applet:** Java applets were programs that were embedded in other applications, typically in a Web page displayed in a web browser. The Java applet API is now deprecated since Java 8 in 2017.
- **Servlet:** Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. Servlets are server-side Java EE components that generate responses (typically HTML pages) to requests (typically HTTP requests) from clients.
- **JavaServer Pages:** JavaServer Pages (JSP) are server-side Java EE components that generate responses, typically HTML pages, to HTTP requests from clients. JSPs embed Java code in an HTML page by using the special delimiters <% and %>. A JSP is compiled to a Java servlet, a Java application in its own right, the first time it is accessed. After that, the generated servlet creates the response.

- **Swing application:** Swing is a graphical user interface library for the Java SE platform. It is possible to specify a different look and feel through the pluggable look and feel system of Swing. Clones of Windows, GTK+, and Motif are supplied by Sun. Apple also provides an Aqua look and feel for macOS. Where prior implementations of these looks and feels may have been considered lacking, Swing in Java SE 6 addresses this problem by using more native GUI widget drawing routines of the underlying platforms.
- **JavaFX application:** JavaFX is a software platform for creating and delivering desktop applications, as well as rich Internet applications (RIAs) that can run across a wide variety of devices. JavaFX is intended to replace Swing as the standard GUI library for Java SE, but both will be included for the foreseeable future. JavaFX has support for desktop computers and web browsers on Microsoft Windows, Linux, and macOS. JavaFX does not have support for native OS look and feels.
- **Generics:** In 2004, generics were added to the Java language, as part of J2SE 5.0. Prior to the introduction of generics, each variable declaration had to be of a specific type. For container classes, for example, this is a problem because there is no easy way to create a container that accepts only specific types of objects. Either the container operates on all subtypes of a class or interface, usually Object, or a different container class has to be created for each contained class. Generics allow compile-time type checking without having to create many container classes, each containing almost identical code. In addition to enabling more efficient code, certain runtime exceptions are prevented from occurring, by issuing compile-time errors. If Java prevented all runtime type errors (ClassCastExceptions) from occurring, it would be type safe.
  In 2016, the type system of Java was proven unsound.

3.1.2 Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++.

3.1.2.1 Features of Android

- Beautiful UI: Android OS basic screen provides a beautiful and intuitive user interface.
- Connectivity: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.
- Storage: SQLite, a lightweight relational database, is used for data storage purposes.
- Media support: H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.
- Messaging: SMS and MMS
- Web browser: Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.
- Multi-touch: Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.
- Multi-tasking: Users can jump from one task to another and same time various application can run simultaneously.

- Resizable widgets: Widgets are resizable, so users can expand them to show more content or shrink them to save space.
- Multi-Language: Supports single direction and bi-directional text.
- GCM: Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.
- Wi-Fi Direct: A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection
- Android Beam: A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go and Android Studio 3.0 or later supports Kotlin and all Java 7 language features and a subset of Java 8 language features that vary by platform version. External projects backport some Java 9 features. While IntelliJ that Android Studio is built on supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

3.1.2.2 Versions

The following is a list of Android Studio's major releases:

Figure 1: Android Versions

3.1.2.3 Android Applications

- Android applications are usually developed in the Java language using the Android Software Development Kit. Once developed, Android applications can be packaged easily and sold out either through a store such as Google Play, SlideME, Opera Mobile Store, Mobango, F-droid and the Amazon Appstore.
- Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide.

Figure 2: Android Applications

3.1.2.4 System Requirements

The Android Emulator has additional requirements beyond the basic system requirements for Android Studio, which are described below:

- SDK Tools 26.1.1 or higher
- 64-bit processor
- Windows: CPU with UG (unrestricted guest) support
- HAXM 6.2.1 or later (HAXM 7.2.0 or later recommended).
- The use of hardware acceleration has additional requirements on Windows and Linux:
- Intel processor on Windows or Linux: Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality
- AMD processor on Linux: AMD processor with support for AMD Virtualization (AMD-V) and Supplemental Streaming SIMD Extensions 3 (SSSE3)
- AMD processor on Windows: Android Studio 3.2 or higher and Windows 10 April 2018 release or higher for Windows Hypervisor Platform (WHPX) functionality.

4. LIBRARIES USED

4.1. Text to Speech

Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech computer or speech synthesizer, and can be implemented in software or hardware products. A text-to-speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech.

Text-to-speech synthesis -TTS - is the automatic conversion of a text into speech that resembles, as closely as possible, a native speaker of the language reading that text. Text-to speech synthesizer (TTS) is the technology which lets computer speak to you. The TTS system gets the text as the input and then a computer algorithm which called TTS engine analyses the text, pre-processes the text and synthesizes the speech with some mathematical models. The TTS engine usually generates sound data in an audio format as the output. The text-to-speech (TTS) synthesis procedure consists of two main phases. The first is text analysis, where the input text is transcribed into a phonetic or some other linguistic representation, and the second one is the generation of speech waveforms, where the output is produced from this phonetic and prosodic information. These two phases are usually called high and low-level synthesis. The input text might be for example data from a word processor, standard ASCII from email, a mobile text-message, or scanned text from a newspaper. The character string is then pre-processed and analyzed into phonetic representation which is usually a string of phonemes with some additional information for correct intonation, duration, and stress. Speech sound is finally generated with the low-level synthesizer by the information from high-level one.

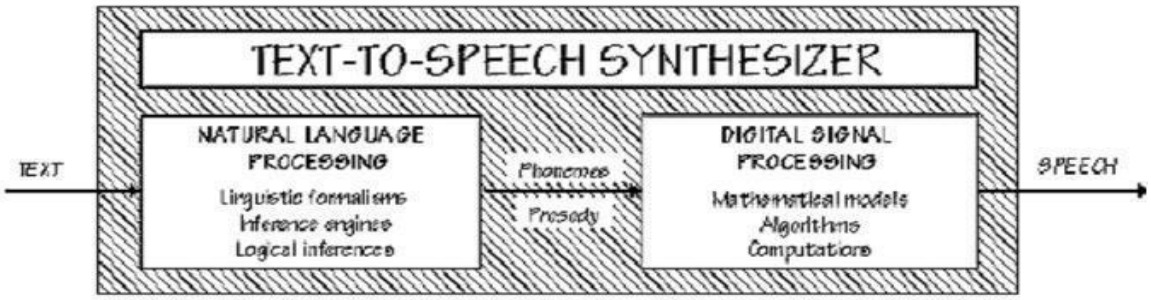

Figure 3: General Functional diagram of Text To Speech(TTS)

The text-to-speech (TTS) synthesis procedure consists of two main phases. The first is text analysis, where the input text is transcribed into a phonetic or some other linguistic representation, and the second one is the generation of speech waveforms, where the output is produced from this phonetic and prosodic information. These two phases are usually called high and low-level synthesis. A simplified version of this procedure is presented in figure 1 below. The input text mightbe for example data from a word processor, standard ASCII
from e-mail, a mobile text-message, or scanned text from a newspaper. The character string is then pre-processed and analyzed into phonetic representation which is usually a string of phonemes with some additional information for correct intonation, duration, and stress. Speech sound is finally generated with the low-level synthesizer by the information from high-level one. The artificial production of speech-like sounds has a long history, with documented mechanical attempts dating to the eighteenth century.

4.1.1. Overview of Speech Synthesis

Speech synthesis can be described as artificial production of human speech. A computer system used for this purpose is called a speech synthesizer, and can be implemented in software or hardware. A text-to-speech (TTS) system converts normal language text into speech. Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database. Systems differ in the size of the stored speech units; a system that stores phones or diphones provides the largest output range, but may lack clarity. For specific usage domains, the storage of entire words or sentences allows for high-quality output. Alternatively, a synthesizer can incorporate a model of the vocal tract and other human voice characteristics to create a completely "synthetic" voice output.

The quality of a speech synthesizer is judged by its similarity to the human voice and by its ability to be understood. An intelligible text-to-speech program allows people with visual impairments or reading disabilities to listen to written works on a home computer.

A text-to-speech system (or "engine") is composed of two parts: a front-end and a back-end. The front-end has two major tasks. First, it converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. This process is often called text normalization, pre- processing, or tokenization. The front-end then assigns phonetic transcriptions to each word, and divides and marks the text into prosodic units, like phrases, clauses, and sentences. The process of assigning phonetic transcriptions to words is called text-to-phoneme or grapheme-to-phoneme conversion. Phonetic transcriptions and prosody information together make up the symbolic linguistic representation that is output by the front-end. The back-end—often referred to as the synthesizer—then converts the symbolic linguistic representation into sound. In certain systems, this part includes the computation of the target prosody (pitch contour, phoneme durations), which is then imposed on the output speech.

4.1.2. Structure of A Text-To-Speech Synthesizer System

Text-to-speech synthesis takes place in several steps. The TTS systems get a text as input, which it first must analyze and then transform into a phonetic description. Then in a further step it generates the prosody. From the information now available, it can produce a speech signal. The structure of the text-to-speech synthesizer can be broken down into major modules:

- **Natural Language Processing (NLP) module:** It produces a phonetic transcription of the text read, together with prosody.
- **Digital Signal Processing (DSP) module:** It transforms the symbolic information it receives from NLP into audible and intelligible speech. The major operations of the NLP module are as follows:
- **Text Analysis:** First the text is segmented into tokens. The token-to-word conversion creates the orthographic form of the token. For the token “Mr” the orthographic form “Mister” is formed by expansion, the token “12” gets the orthographic form “twelve” and “1997” is transformed to “nineteen ninety seven”.

- **Application of Pronunciation Rules:** After the text analysis has been completed, pronunciation rules can be applied. Letters cannot be transformed 1:1 into phonemes because correspondence is not always parallel. In certain environments, a single letter can correspond to either no phoneme (for example, “h” in “caught”) or several phoneme (“m” in “Maximum”). In addition, several letters can correspond to a single phoneme (“ch” in “rich”).

4.2. Optical character Recognition

Optical character recognition or optical character reader (OCR) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast). Widely used as a form of data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common, and with support for a variety of digital image file format inputs. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

4.2.1. Types

Optical character recognition (OCR) – targets typewritten text, one glyph or character at a time.

Optical word recognition – targets typewritten text, one word at a time (for languages that use a space as a word divider). (Usually just called "OCR".)

Intelligent character recognition (ICR) – also targets handwritten printscript or cursive text one glyph or character at a time, usually involving machine learning.

Intelligent word recognition (IWR) – also targets handwritten printscript or cursive text, one word at a time. This is especially useful for languages where glyphs are not separated in cursive script.

OCR is generally an "offline" process, which analyses a static document. There are cloud based services which provide an online OCR API service. Handwriting movement analysis can be used as input to handwriting recognition. Instead of merely using the shapes of glyphs and words, this technique is able to capture motions, such as the order in which segments are drawn, the direction, and the pattern of putting the pen down and lifting it. This additional information can make the end-to-end process more accurate. This technology is also known as "on-line character recognition", "dynamic character recognition", "real-time character recognition", and "intelligent character recognition".

4.2.2. Techniques

Pre-processing

OCR software often "pre-processes" images to improve the chances of successful recognition. Techniques include:

- De-skew – If the document was not aligned properly when scanned, it may need to be tilted a few degrees clockwise or counterclockwise in order to make lines of text perfectly horizontal or vertical.
- Despeckle – remove positive and negative spots, smoothing edges
- Binarisation – Convert an image from color or greyscale to black-and-white (called a "binary image" because there are two colors). The task of binarisation is performed as a simple way of separating the text (or any other desired image component) from the background. The task of binarisation itself is necessary since most commercial recognition algorithms work only on binary images since it proves to be simpler to do so. In addition, the effectiveness of the binarisation step influences to a significant extent the quality of the character recognition stage and the careful decisions are made in the choice of the binarisation employed for a given input image type; since the quality of the binarisation method employed to obtain the binary result depends on the type of the input image (scanned document, scene text image, historical degraded document etc.).
- Line removal – Cleans up non-glyph boxes and lines
- Layout analysis or "zoning" – Identifies columns, paragraphs, captions, etc. as distinct blocks. Especially important in multi-column layouts and tables.
- Line and word detection – Establishes baseline for word and character shapes, separates words if necessary.
- Script recognition – In multilingual documents, the script may change at the level of the words and hence, identification of the script is necessary, before the right OCR can be invoked to handle the specific script.

- Character isolation or "segmentation" – For per-character OCR, multiple characters that are connected due to image artifacts must be separated; single characters that are broken into multiple pieces due to artifacts must be connected.
- Normalize aspect ratio and scale.

Segmentation of fixed-pitch fonts is accomplished relatively simply by aligning the image to a uniform grid based on where vertical grid lines will least often intersect black areas. For proportional fonts, more sophisticated techniques are needed because whitespace between letters can sometimes be greater than that between words, and vertical lines can intersect more than one character.

Text recognition

There are two basic types of core OCR algorithm, which may produce a ranked list of candidate characters.

Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is also known as "pattern matching", "pattern recognition", or "image correlation". This relies on the input glyph being correctly isolated from the rest of the image, and on the stored glyph being in a similar font and at the same scale. This technique works best with typewritten text and does not work well when new fonts are encountered. This is the technique the early physical photocell-based OCR implemented, rather directly.

Feature extraction decomposes glyphs into "features" like lines, closed loops, line direction, and line intersections. The extraction features reduces the dimensionality of the representation and makes the recognition process computationally efficient. These features are compared with an abstract vector-like representation of a character, which might reduce to one or more glyph prototypes. General techniques of feature detection in computer vision are applicable to this type of OCR, which is commonly seen in "intelligent" handwriting recognition and indeed most modern OCR software. Nearest neighbor classifiers such as the k-nearest neighbors algorithm are used to compare image features with stored glyph features and choose the nearest match.

Software such as Cuneiform and Tesseract use a two-pass approach to character recognition. The second pass is known as "adaptive recognition" and uses the letter shapes recognized with high confidence on the first pass to recognize better the remaining letters on the second pass.

This is advantageous for unusual fonts or low-quality scans where the font is distorted (e.g. blurred or faded).

Modern OCR software like for example OCRopus or Tesseract uses neural networks which were trained to recognize whole lines of text instead of focusing on single characters.

The OCR result can be stored in the standardized ALTO format, a dedicated XML schema maintained by the United States Library of Congress. Other common formats include hOCR and PAGE XML.

For a list of optical character recognition software see Comparison of optical character recognition software.

Post-processing

OCR accuracy can be increased if the output is constrained by a lexicon – a list of words that are allowed to occur in a document. This might be, for example, all the words in the English language, or a more technical lexicon for a specific field. This technique can be problematic if the document contains words not in the lexicon, like proper nouns. Tesseract uses its dictionary to influence the character segmentation step, for improved accuracy.

The output stream may be a plain text stream or file of characters, but more sophisticated OCR systems can preserve the original layout of the page and produce, for example, an annotated PDF that includes both the original image of the page and a searchable textual representation.

"Near-neighbor analysis" can make use of co-occurrence frequencies to correct errors, by noting that certain words are often seen together. For example, "Washington, D.C." is generally far more common in English than "Washington DOC".

Knowledge of the grammar of the language being scanned can also help determine if a word is likely to be a verb or a noun, for example, allowing greater accuracy.

The Levenshtein Distance algorithm has also been used in OCR post-processing to further optimize results from an OCR API.

Application-specific optimizations

In recent years, the major OCR technology providers began to tweak OCR systems to deal more efficiently with specific types of input. Beyond an application-specific lexicon, better performance may be had by taking into account business rules, standard expression, or rich information contained in color images. This strategy is called "Application-Oriented OCR" or "Customized OCR", and has been applied to OCR of license plates, invoices, screenshots, ID cards, driver licenses, and automobile manufacturing.

The New York Times has adapted the OCR technology into a proprietary tool they entitle, Document Helper that enables their interactive news team to accelerate the processing of documents that need to be reviewed. They note that it enables them to process what amounts to as many as 5,400 pages per hour in preparation for reporters to review the contents.

4.2.3. Applications

OCR engines have been developed into many kinds of domain-specific OCR applications, such as receipt OCR, invoice OCR, check OCR, legal billing document OCR.

They can be used for:

- Data entry for business documents, e.g. check, passport, invoice, bank statement and receipt
- Automatic number plate recognition
- In airports, for passport recognition and information extraction
- Automatic insurance documents key information extraction
- Traffic sign recognition
- Extracting business card information into a contact list
- More quickly make textual versions of printed documents, e.g. book scanning for Project Gutenberg
- Make electronic images of printed documents searchable, e.g. Google Books
- Converting handwriting in real time to control a computer (pen computing)
- Defeating CAPTCHA anti-bot systems, though these are specifically designed to prevent OCR. The purpose can also be to test the robustness of CAPTCHA anti-bot systems.

- Assistive technology for blind and visually impaired users
- Writing the instructions for vehicles by identifying CAD images in a database that are appropriate to the vehicle design as it changes in real time.
- Making scanned documents searchable by converting them to searchable PDFs

5. MODULES

5.1 Text to Speech

Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech computer or speech synthesizer, and can be implemented in software or hardware products. A text-to-speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech.

5.2 Speech to Text

A speech-to-text reporter (STTR), also known as a captioner, is a person who listens to what is being said and inputs it, word for word (verbatim), using an electronic shorthand keyboard or speech recognition software and a CAT software system. The latest in speech-to-text reporters are called Voice writers.

5.3 Notification Reader

A notification reader helps you to read the notifications when it pops up irrespective whether the app is active or not. It reads the message and also the person who sent the message. This is done using a service named Notification initListener in android Studio.

5.4 Image Processesing

Image Processessing is the given image is converted into speech which means when you open the app and camera is provided and when you click the image it recognizes the image and tells you the name of the image.

5.5 File Uploader

File Uploader helps you to upload the files. The files that are uploaded will be converted into speech.

6. IMPLEMENTATION

Text to Speech

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import java.util.Locale;
public class TextToSpeechActivity extends AppCompatActivity {
   TextToSpeech t1;
   EditText ed1;
   Button b1;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_text_to_speech);
      ed1=(EditText)findViewById(R.id.T2STv);
      b1=(Button)findViewById(R.id.T2SBtn);
      getSupportActionBar().setTitle("TextToSpeech");
      getSupportActionBar().setDisplayHomeAsUpEnabled(true);
      getSupportActionBar().setDisplayShowHomeEnabled(true);
      t1=new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {
         @Override
         public void onInit(int status) {
            if(status != TextToSpeech.ERROR) {
               t1.setLanguage(Locale.ENGLISH);
            }
         }
      });

      b1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            String toSpeak = ed1.getText().toString();
Toast.makeText(getApplicationContext(),toSpeak,Toast.LENGTH_SHORT).show();
            spk(toSpeak);
         }
      });
   }
```

```
public void onPause(){

        if(t1 !=null){
            t1.stop();
            t1.shutdown();
        }
        super.onPause();
    }
    @Override
    public boolean onSupportNavigateUp() {
        onBackPressed();
        return true;
    }

    void spk(String msg){
        t1.speak(msg, TextToSpeech.QUEUE_FLUSH, null);

    }
}
```

Speech to Text

```
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.speech.RecognizerIntent;
import android.view.View;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import java.util.ArrayList;
import java.util.Locale;
public class SpeechToTextActivity extends AppCompatActivity {
    private static final int REQUEST_CODE_SPEECH_INPUT = 1000;
    TextView mTextTv;
    ImageView mVoiceBtn;
    @Override
protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_speech_to_text);
        mTextTv =findViewById(R.id.S2TTv);


        getSupportActionBar().setTitle("SpeechToText");
        mVoiceBtn=findViewById(R.id.S2TImg);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        mVoiceBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                speak();

            }
        });
    }

    private void speak() {
        Intent intent=new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,RecognizerIntent.LANGUA
GE_MODEL_FREE_FORM);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE,Locale.getDefault());
intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Hi speak something");

        try{
            startActivityForResult(intent,REQUEST_CODE_SPEECH_INPUT);

        }
        catch(Exception e){
            Toast.makeText(this,""+e.getMessage(),Toast.LENGTH_SHORT).show();

        }
    }
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data)

{
        super.onActivityResult(requestCode, resultCode, data);
        switch(requestCode){
            case REQUEST_CODE_SPEECH_INPUT:{
                if(resultCode==RESULT_OK && null!=data){
                    ArrayList<String>
result=data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                    mTextTv.setText(result.get(0));
                }
```

```
            break;
        }
    }
  }
  @Override
  public boolean onSupportNavigateUp() {
    onBackPressed();
    return true;
  }
}
```

File to Speech

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.app.Activity;
import android.content.ContentUris;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.provider.DocumentsContract;
import android.provider.MediaStore;
import android.speech.tts.TextToSpeech;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
```

```
import android.widget.Toast;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.util.Scanner;
public class FileToSpeechActivity extends AppCompatActivity {
String text;
TextView tv;
private String TAG ="";
private int request_code =1, FILE_SELECT_CODE =101;
Button upload,submit;
ImageView iv;
private int REQUEST_CAMERA = 0;
private int PICK_IMAGE_REQUEST = 1;
private Bitmap bitmap;
private Uri filePath;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_file_to_speech);
        tv = findViewById(R.id.F2STv);
        upload = findViewById(R.id.F2SUplad);
    //    submit = findViewById(R.id.F2SBtn);
        upload.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (ContextCompat.checkSelfPermission(getApplicationContext(),
android.Manifest.permission.READ_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(getApplicationContext(),
android.Manifest.permission.READ_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED) {

                }
                showFileChooser();
            }
        });
```

```
//        submit.setOnClickListener(new View.OnClickListener() {
//            @Override

//            public void onClick(View view) {
//
//            }
//        });

    }
    private void showFileChooser() {
        Intent intent = new Intent();
        intent.setType("*/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(Intent.createChooser(intent, "Select File"),
PICK_IMAGE_REQUEST);
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == PICK_IMAGE_REQUEST && resultCode == RESULT_OK && data
!= null && data.getData() != null) {

            filePath = data.getData();
            try {

                InputStream inputStream = getContentResolver().openInputStream(filePath);
                Scanner input = new Scanner(inputStream);
                StringBuffer br = new StringBuffer();

                while(input.hasNext()){
                    String line = input.nextLine();
                    br.append(line);

                }
                tv.setText(br.toString());
                MyService.mTts.speak(br.toString(), TextToSpeech.QUEUE_FLUSH, null);



                // bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), filePath);
                //iv.setImageBitmap(bitmap);
            }
```

```
 catch (Exception e) {

            e.printStackTrace();
        }
    }


  }


}
```

Notification Reader

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.media.AudioManager;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.telephony.SmsMessage;
import android.widget.Toast;
import java.util.HashMap;
import java.util.Locale;
public class Mode extends BroadcastReceiver {
   TextToSpeech t1;
   @Override
   public void onReceive(Context context, Intent intent) {
      try {

         t1=new TextToSpeech(context.getApplicationContext(), new
TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int status) {
               if(status != TextToSpeech.ERROR) {
                  t1.setLanguage(Locale.ENGLISH);
               }
            }
         });

         Bundle bundle = intent.getExtras();
         //---retrieve the SMS message received---
         Object[] messages = (Object[]) bundle.get("pdus");
```

```
        SmsMessage[] sms = new SmsMessage[messages.length];
        for (int n = 0; n < messages.length; n++) {

           sms[n] = SmsMessage.createFromPdu((byte[]) messages[n]);
        }

        for (SmsMessage x : sms) {
           String mobile = x.getOriginatingAddress();
           String msgbody = x.getMessageBody().toString().trim().toUpperCase();
           String msg = "hi you got a text message from "+mobile+" and the message is
"+msgbody;

           Toast.makeText(context, msg, Toast.LENGTH_LONG).show();

           MyService.mTts.speak(msg, TextToSpeech.QUEUE_FLUSH, null);

        }
      }
catch (Exception e){

      }
   }

   public void pause(int duration){
      t1.playSilence(duration, TextToSpeech.QUEUE_ADD, null);
   }
}
```

Image to Speech

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.util.Log;
import android.util.SparseArray;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.TextView;
import com.google.android.gms.vision.CameraSource;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.text.TextBlock;
import com.google.android.gms.vision.text.TextRecognizer;
```

```
import java.io.IOException;
public class ImageToSpeechActivity extends AppCompatActivity {
   private static final String TAG = "";
   CameraSource mCameraSource;
   SurfaceView mCameraView;
   int requestPermissionID;
   TextView mTextView;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_image_to_speech);
      mCameraView = findViewById(R.id.surfaceView);
      mTextView = findViewById(R.id.text_view);
      startCameraSource();
   }

   private void startCameraSource() {
      final TextRecognizer textRecognizer = new
TextRecognizer.Builder(getApplicationContext()).build();
      if (!textRecognizer.isOperational()) {
         Log.w(TAG, "Detector dependencies not loaded yet");
      } else {
         mCameraSource = new CameraSource.Builder(getApplicationContext(), textRecognizer)
               .setFacing(CameraSource.CAMERA_FACING_BACK)
               .setRequestedPreviewSize(1280, 1024)
               .setAutoFocusEnabled(true)
               .setRequestedFps(2.0f)
               .build();


         mCameraView.getHolder().addCallback(new SurfaceHolder.Callback() {
            @Override
            public void surfaceCreated(SurfaceHolder holder) {
               try {

                  if (ActivityCompat.checkSelfPermission(getApplicationContext(),
                        Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED) {

ActivityCompat.requestPermissions(ImageToSpeechActivity.this,
                           new String[]{Manifest.permission.CAMERA},
                           requestPermissionID);
                     return;
                  }
```

```
                mCameraSource.start(mCameraView.getHolder());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            mCameraSource.stop();
        }
    });

    textRecognizer.setProcessor(new Detector.Processor<TextBlock>() {
        @Override
        public void release() {
        }


        @Override
        public void receiveDetections(Detector.Detections<TextBlock> detections) {
            final SparseArray<TextBlock> items = detections.getDetectedItems();
            if (items.size() != 0 ){

                mTextView.post(new Runnable() {
                    @Override
                    public void run() {
                        StringBuilder stringBuilder = new StringBuilder();

for(int i=0;i<items.size();i++){

                            TextBlock item = items.valueAt(i);
                            stringBuilder.append(item.getValue());
                            stringBuilder.append("\n");
                        }
                        mTextView.setText(stringBuilder.toString());
                        MyService.mTts.speak(stringBuilder.toString(),
TextToSpeech.QUEUE_FLUSH, null);
                    }
                });
            }
        }
    });
  }
```

7. RESULTS

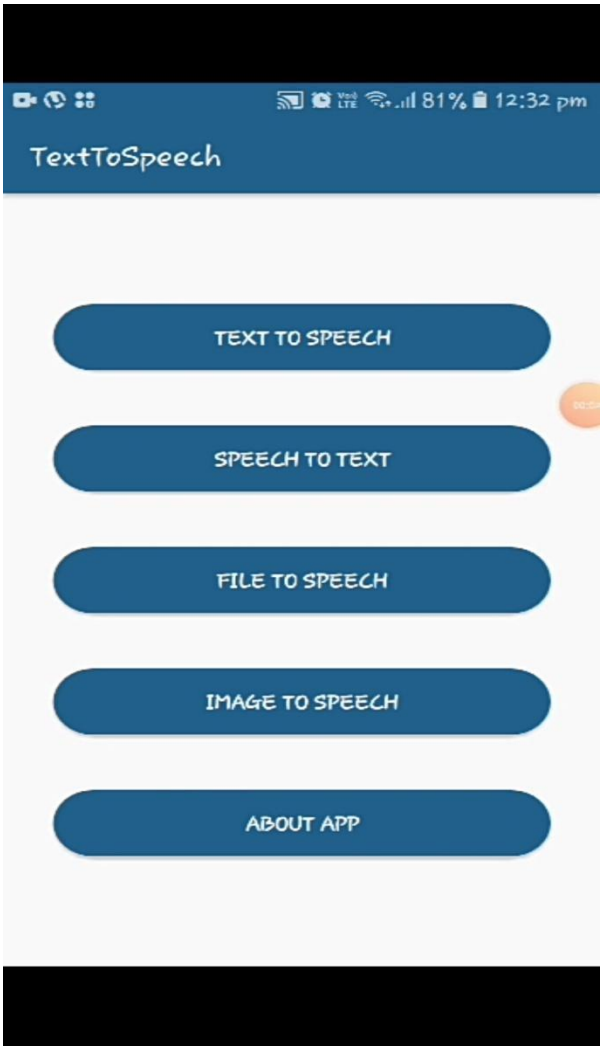

Figure 4: Home Page

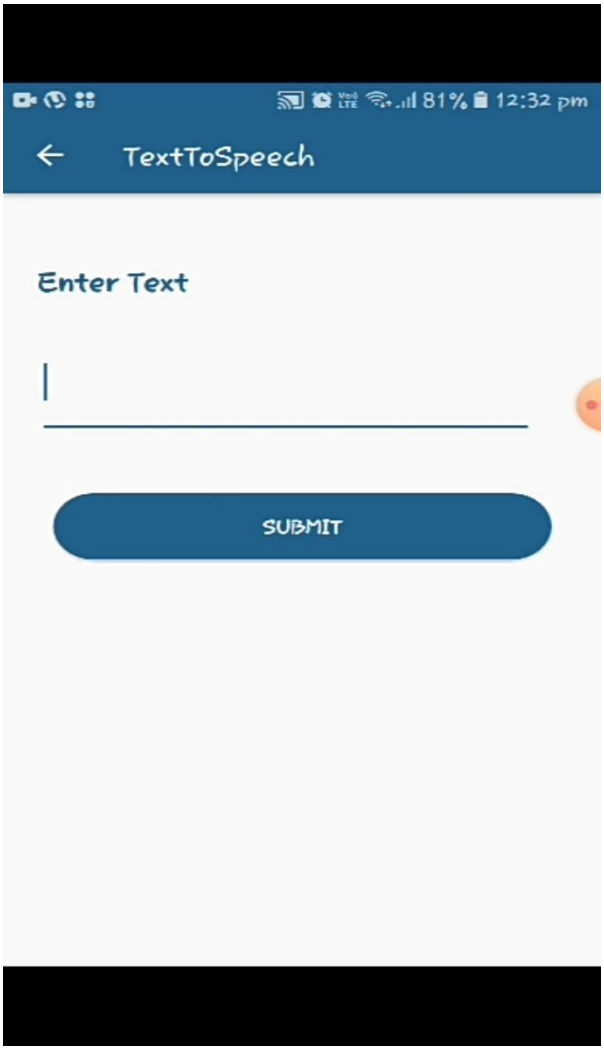

Figure 5: Text To Speech

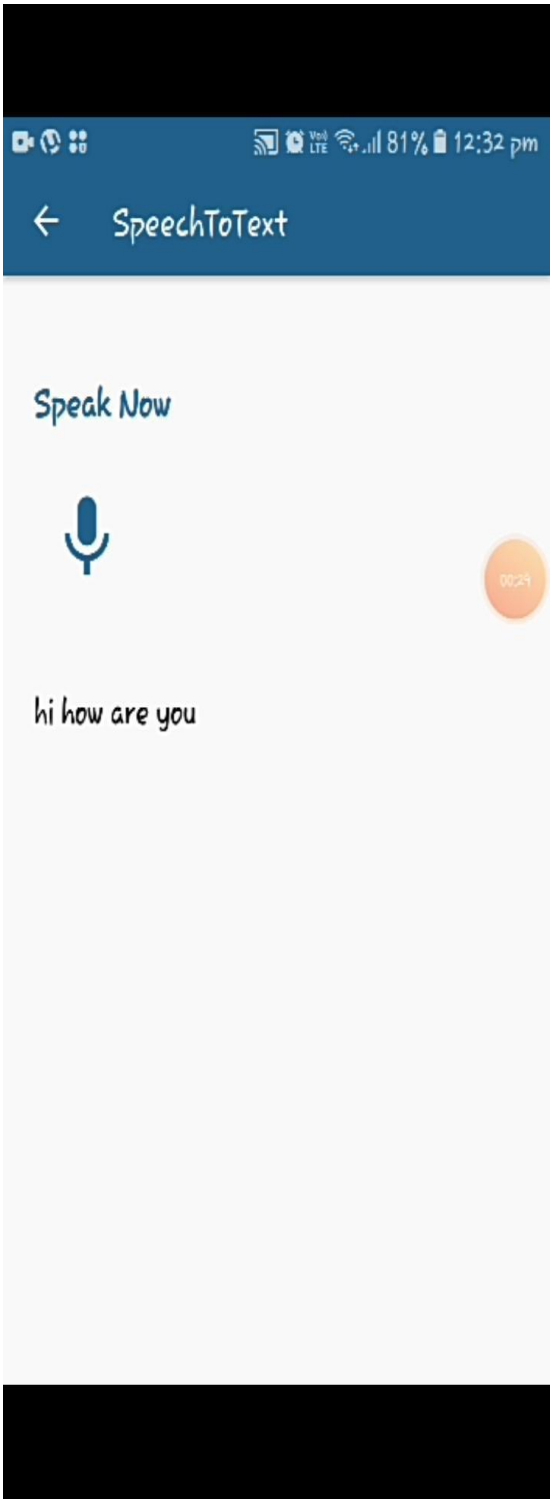

Figure6: Speech to Text

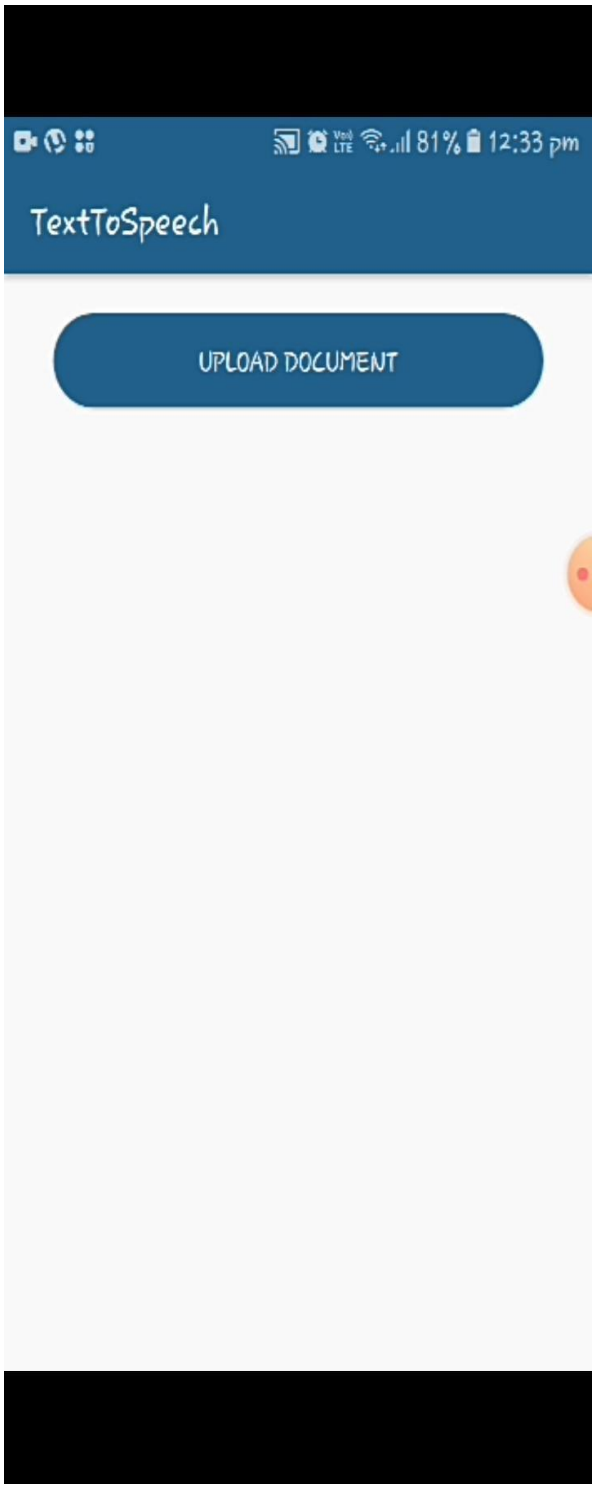

Figure7:FileUploader

Figure 8: Image Processessing

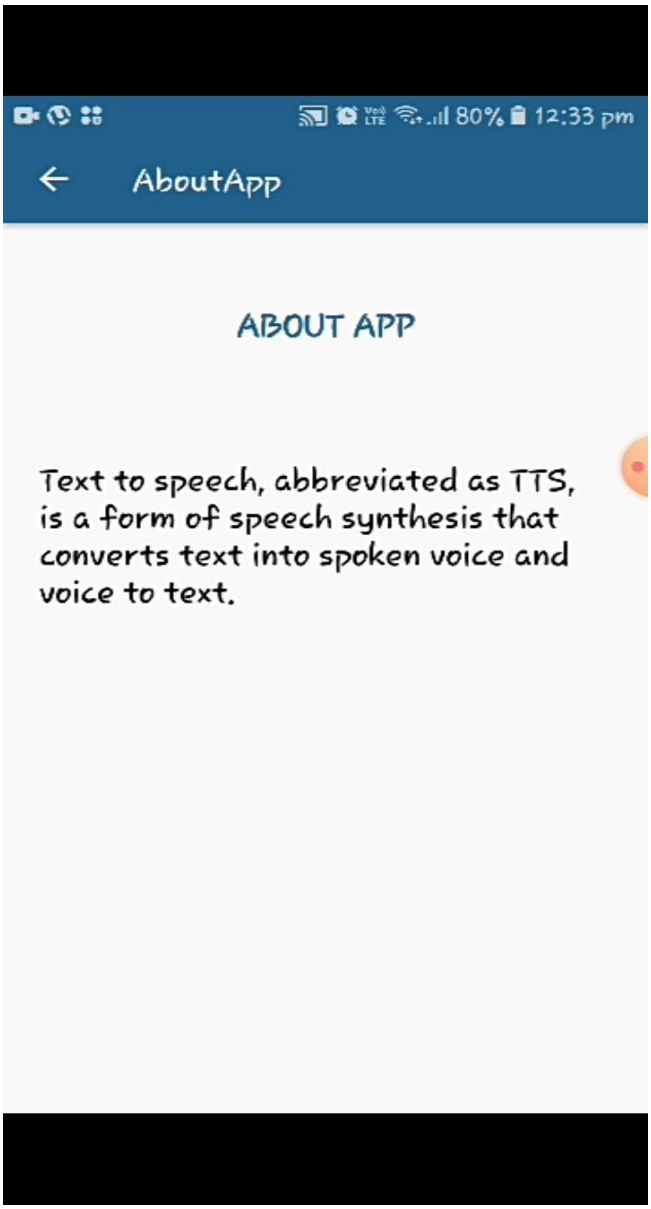

Figure 9: About the App

8. CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

Text to speech and speech to text synthesis is a rapidly growing aspect of computer technology and is increasingly playing a more important role in the way we interact with the system and interfaces across a variety of platforms. We have developed a very simple and attractive graphical user interface which allows the user to type text to convert it into speech, converts the speech to text, allows the user to upload text files, read out the text from the image and the android read out the text messages the user receives.

8.2 Future Scope

In future, we plan to work on the implementation of a text to speech system on other platforms, such as telephony systems, ATM machines, video games and any other platforms where text to speech technology would be an added advantage and increase functionality.

9. REFERENCES

[1] https://core.ac.uk/download/pdf/83592918.pdf

[2].https://www.tutorialspoint.com/android/android_text_to_speech.htm

[3].https://www.researchgate.net/publication/282268546_Text_to_Speech_Conversion_System_using_OCR

[4].https://www.techradar.com/in/news/the-best-free-text-to-speech-software

[5].https://www.engpaper.com/voice-recognition.htm

[6].https://www.researchgate.net/publication/313389686_Text_to_speech_conversion