A Project Report

# Improving Text Extraction Accuracy with Image Preprocessing

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

**BACHELOR OF ENGINEERING**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

***by***

**Komali Beeram (160118733067)**

**Soumya Vemuri (160118733081)**

**Department of Computer Science and Engineering,**
**Chaitanya Bharathi Institute of Technology (Autonomous),**
**(Affiliated to Osmania University, Hyderabad)**
**Hyderabad, TELANGANA (INDIA) –500 075**
**[2021-2022]**

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)
Kokapet ( Village), Gandipet, Hyderabad, Telangana-500075. www.cbit.ac.in
ISO Certified 9001:2015

COMMITTED TO RESEARCH, INNOVATION AND EDUCATION 43 years

# CERTIFICATE

This is to certify that the project titled **"Improving Text Extraction Accuracy with Image Preprocessing"** is the bonafide work carried out by Komali Beeram (160118733067) and Soumya Vemuri (160118733081), students of B.E.(CSE) of Chaitanya Bharathi Institute of Technology(A), Hyderabad, affiliated to Osmania University, Hyderabad, Telangana(India) during the academic year 2021-2022, submitted in partial fulfillment of the requirements for the award of the degree in Bachelor of Engineering (Computer Science and Engineering) and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

**Project Supervisor**
Dr. T. Sridevi
Associate Professor,
Department of Computer Science and Engineering,
Chaitanya Bharathi Institute of Technology

**Head of Department**
Dr. Y. Ramadevi,
Professor,
Department of Computer Science and Engineering,
Chaitanya Bharathi Institute of Technology

Place: Hyderabad
Date: 27th May, 2022

# DECLARATION

We hereby declare that the project entitled, “Improving Text Accuracy with Image Preprocessing” submitted for the B.E. (CSE) degree is our original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

**Komali Beeram**
160118733067
Department of Computer Science and Engineering,
Chaitanya Bharathi Institute of Technology

**Soumya Vemuri**
160118733081
Department of Computer Science and Engineering,
Chaitanya Bharathi Institute of Technology

Place: Hyderabad
Date: 27th May, 2022

# ABSTRACT

Digitization allows us to immortalize a physical entity by creating a digital representation of it on our devices. It saves us time in manually sifting through physical storage units such as albums and notebooks and provides us with programs to manage and secure our data. We often take images of Receipts or Invoices, Identity Cards, and nutritional labels to save a copy of their details. This can be taken a step further by automating the process of information extraction and documentation.

Advancements in computer vision have provided us with the expertise to create tools for text detection and extraction. But it is still an ongoing challenge because documents with unstructured layouts, poor image quality, and noise around the text yield very low accuracy in text extraction results. Conquering this challenge would require the image to be highly enhanced through pre-processing techniques such as Brightness Correction, Contour Detection, Skewness Correction, Morphology, and Binarization. A mechanism made from the best combination of image pre-processing techniques prior to text extraction can improve text accuracy to a large extent.

# ACKNOWLEDGEMENT

The success and the outcome of this project required a lot of guidance and assistance from many people and sources, and we are extremely fortunate to have received this all along the completion of the project.

We would like to express our special thanks of gratitude to our supervisor and mentor, Dr. T. Sridevi, Associate Professor, Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, for her invaluable guidance, moral support, and patience while suggesting the outlines of this project, and time to time counselling and advice. We thank for her overall support.

We also convey our sincere thanks to the Head of Department of Computer Science and Engineering, Dr. Y. Ramadevi, for her assistance.

We also thank our college professors, Chaitanya Bharathi Institute of Technology, for giving us the opportunity to work on an innovative idea under the course Projects.

At last, we would like to thank our families and friends for their constant support and guidance from time to time in making this project well above simplicity and into something concrete, despite their busy schedules.

# LIST OF FIGURES

# TABLE OF CONTENTS

# 1. INTRODUCTION

In today's day and age, an increase in demand for digitization has fueled a massive growth in technology and communication and the use of printed materials such as books and papers has significantly reduced. It is easier to organize digitized data and analyze them for various purposed with many advanced techniques like artificial intelligence etc. To translate physical and handwritten documents into digital copies, optical character recognition (OCR) has come into the sight of researchers and since its first advent it has undergone significant changes in methodology and made considerable progress towards its goal of text detection and extraction.

## 1.1. Problem Statement

Historical OCR engines have their accuracy lying between 70-80% for a high-quality image at page level. That means in a page of 100 words 70-80 words are accurate. This will lead to significant inaccuracies if used on a large volume of sensitive documents.

Through this project we aim to improve the accuracy of the result by implementing an ideal combination of image preprocessing techniques prior to text extraction. The aim is to extend existing pipelines to improve the accuracy by; improving the quality of the image and using Natural Language Processing to extract information from raw text.

## 1.2. Methodology

I. **Document Scan and File Upload**

User scans their document and uploads it to the system interface. Documents include invoices, receipts, nutrition labels, book covers etc.

II. **Image Preprocessing Pipeline**

Scanned documents undergo the best combination of image preprocessing techniques for maximum image text enhancement. Image Preprocessing Techniques include

a. Brightness Correction
Brightness correction corrects light variation produced by surface relief or document curvature. When a thick book is scanned, the shadow of the binding will appear on the image. This technique allows obtaining light uniformity and eliminate such shadows, whether vertical or horizontal.

b. Contour Detection
Contours can be explained simply as a curve joining all the continuous points along the boundary, having the same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition. For better accuracy, binary images are used.

c. Scaling of Image

   Ensure that the images are scaled to the right size which usually is of at least 300 Dots Per Inch. Keeping DPI lower than 200 will give unclear and incomprehensible results while keeping the DPI above 600 will unnecessarily increase the size of the output file without improving the quality of the file. Thus, a DPI of 300 works best for this purpose.

d. Skewness Correction

   When the optical axis of the camera is not perpendicular to the text plane, Perspective distortion occurs. Text boundaries lose rectangular shapes and characters distort, decreasing the performance of recognition models trained on undistorted samples.

e. Noise Removal

   Noise can drastically reduce the overall quality of the OCR process. It can be present in the background or foreground and can result from poor scanning or the poor original quality of the data.

f. Binarization

   This step converts a multicolored image (RGB) to a black and white image. There are several algorithms to convert a color image to monochrome image, ranging from simple thresholding to more sophisticated zonal analysis.

g. Contrast Correction

   Low contrast can result in poor OCR. Increase the contrast and density before carrying out the OCR process. Increasing the contrast between the text/image and its background brings out more clarity in the output.

**III. Text Extraction**

The image goes through an Optical Character Recognition (OCR) engine that’s been built to recognize printed text in paper documents, handwritten characters, and text elements in the image.

**IV. Information Extraction**

Information Extraction involves extracting meaningful information from raw text data into a structured format.

**V. Text-to-Speech Conversion**

Text-to-speech is a type of assistive technology that reads digital text aloud. Text-to-speech can take words on a computer or other digital device and convert them into audio.

## 1.3. Outline of Report

For performance of recognition, we will simulate the proposed text recognition system to various characters in English language and calculate the recognition rate or accuracy which is expected to improve from the current OCR engines.

## 1.4. Scope of Project

The scope of our project on a grid infrastructure is to provide an efficient and enhanced software tool for the users to perform document image analysis, document processing by reading and recognizing the characters in research, academic, governmental, and business organizations that are having large pool of documented, scanned images. Irrespective of the size of documents and the type of characters in documents, the product is recognizing them, searching them, and processing them faster according to the needs of the environment.

## 1.5. Organization of Report

- Chapter 1: Introduction
  This part summarizes about the project in brief. It includes the problem statement, methodology, and outline of our results and the scope of the project.
- Chapter 2: Literature Survey
  This part briefly describes the overall architecture of text recognition system and provide a brief overview of the existing work carried out in the field of image preprocessing and text recognition.
- Chapter 3: Design of Proposed System
  This part describes the system in the form of a block diagram, the various modules in the overall system, and the theoretical foundation behind our methodology.
- Chapter 4: Implementation of the Proposed System
  This chapter describes the architecture, the pseudo code and the implementation of the system. The dataset and the testing steps are also briefly summarized.
- Chapter 5: Results
  This chapter summarizes the output and the results of the implemented code
- Chapter 6: Conclusion
  In this chapter we conclude the paper by summarizing our observations and describe the system’s limitations and scope for the future

Although the document may be read from front to back for a complete understanding of the project, it was written in sections and hence can be read as such. For an overview of the document and the project itself, refer to Introduction.

# 2. LITERATURE SURVEY

In this chapter we briefly describe the overall architecture of the text recognition system and provide a brief overview of the existing work carried out in the field of image preprocessing and text recognition.

## 2.1. Introduction to the problem domain and terminology

Text recognition has gained a lot of attention in recent years through its entrance into a large arena of applications, and it is a field which is driven by the need to preserve access to the information containing documents in an easier and quicker way. The most convenient way to transfer the information from the paper or books is to scan them, which converts the information into an image, preventing the reuse of the scanned information in the form of a text. One of the popular techniques used for text detection and extraction is Optical Character Recognition (OCR). It converts scanned images of text into editable format.

The process of text recognition starts with capturing the image of the required document, preprocessing it to acquire the desired portion and then segmenting it to extract the text content present in it. The text recognition system can be divided into three modules:

A. **Pre-processing**

   A document is generally scanned and converted into the form of a picture. A picture is the combinations of picture elements which are also known as pixels. At this stage we have the data in the form of image and this image can be further analyzed so that the important information can be retrieved. To improve quality of the input image, few operations are performed to enhance the image, such as noise removal, binarization, and skew correction.

B. **Text Recognition**

   This module can be used for text recognition in output image of pre-processing model and give output data which are in computer understandable form. Hence in this module following techniques are used.

   - Segmentation
     In text recognition module, the segmentation is the most important process. It is done to make the separation between the individual characters of an image.
   - Feature Extraction
     It is the process to retrieve the most important data from the raw data.

To store the different features of a character, different classes are made.

- Classification
  Classification is the process of identifying each character and assigning to it the correct character class, so that texts in images are converted into computer understandable form. This process used extracted feature of text image for classification i.e. input to this stage is output of the feature with stored pattern and find out best matching class for input.

C. **Post-processing**
The output of text recognition module is in the form of text data which is understandable by the computer. So, there is a need to store it into a proper format such as text or word for further use such as editing or searching in that data.

## 2.2. Existing Solutions

Many researchers have contributed and proposed their concepts on text extraction from an image and the retrieval of information. Each solution has its own pros and cons that we have summarized in this subsection.

Rishabh Mittal and Anchal Garg [1] introduced and explained the concept of OCR and the process of extraction by grouping it into majorly six steps: image acquisition, pre-processing, segmentation, feature extraction, classification, and post-processing. The paper reveals that the modern ocr system's preprocessing pipeline is restricted to spatial image filtering, thresholding, noise-removal, and skew detection/correction. Improving components like Scan goals, filtered picture quality, type of printer utilized whether inkjet or laser, the nature of the paper, phonetic complexities, the lopsided brightening, and watermarks can impact the precision of OCR. Hence work can be done on improving the precision of OCR.

Sanjeev Kumar, Mahika Sharma, Kritika Handa, Rishika Jaiswal [2] proposed a novel adaptive algorithm to improve ocr accuracy with advanced image preprocessing using machine learning. Their focus was to reduce the noise of the image solely by scaling the original source image to around 300 DPI which has helped to eliminate the single biggest obstacle of the Tesseract, i.e., Tesseract's computation time of reading images with the highest character dimensions above 20 pixels. However, their algorithm does not cover images with uneven brightness, watermarks, or different fonts.

Sahana K Adyanthaya put forward a paper [3] that presents the various steps taken to recognize text from images. The steps addressed in this paper were Image Preprocessing, Segmentation, Feature Extraction and Classification. The author highlights that the noise present in an image has a major role to play in successful text recognition and that noise removal increases the probability of accurate text recognition and generates more accurate output. The paper mentions that Gaussian filter and mean filter can be used for noise removal, that normalization should be done to ensure uniformity followed by binarization to convert the gray image into a binary image.

Naveen Sankaran and C.V Jawahar [4] proposed a neural network-based framework that operates based on BLSTM-Bidirectional Long Short-Term Memory that allows OCR to work at the word level. It leads to over 20% better results when compared to a regular OCR framework. It uses a method that does not require segmentation, that is one amongst the foremost common reasons for the error. Also, it found an over 9% decrease in character error compared to the more widely available OCR framework.

Work by S. Akopyan, O.V. Belyaeva, T.P. Plechov and D.Y. Turdakov [5] is based on a text extraction pipeline which is used to extract text from varied quality of images obtained from social media. Their work mainly focuses on dividing the input images into various classes and then preprocessing is done depending on the classes. This is followed by text recognition using the OCR engine. The dataset collected from social media is made use of in this work.

Dan Sporici, Elena Cușnir and Costin-Anton Boiangiu [16] underlined Tesseract 4.0 flaws, highly related to the segmentation procedure and proposed an adaptive image preprocessing step guided by a reinforcement learning model, which attempts to minimize the edit distance between the recognized text and the ground truth. This approach has boosted the character-level accuracy of Tesseract 4.0 from 0.134 to 0.616 and the F1 score from 0.163 to 0.729. The model adjusts samples with the purpose of maximizing the overall recognition efficiency without requiring external guidance or knowledge which has a direct benefit of including kernels, which can generate samples that might look unnatural. From a qualitative point of view, the changes are substantial yet not optimal since a reinforcement learning approach does not guarantee that local optimums will be avoided each time and hence the algorithm can get stuck on kernel configurations which will provide inferior results if not enough exploration is performed.

Dr. PL Chitra, and P Bhavani [6], in this paper have studied various images to remove unwanted noise and performed enhancement techniques such as contrast limited adaptive histogram equalization, Laplacian and Harr filtering, unsharp masking, sharpening, high boost filtering and color models then the Clustering algorithms are useful for data logically and extract pattern-analysis, grouping, decision-making, and machine-learning techniques and Segment the regions using binary, K-means and OTSU segmentation algorithm. It classifies the images with the help of SVM and K-Nearest Neighbors (KNN) Classifier to produce good results for those images.

Anupriya Shrivastava, Amudha J.Deepa Gupta and Kshitij Sharma [7] in their work have developed a system based on Convolutional Neural Network and Long ShortTerm Memory. The developed model identifies the texts from images which are horizontal, curved or oriented style. The model has four components. The first component performs feature extraction at the low level. The second component uses a shared convolution approach to extract high level features. Irrelevant features are ignored by the third component. The fourth component predicts the character sequences.

K. Karthick, K.B. Ravindrakumar, R. Francis and S.Ilankannan [8] have discussed the various steps in text detection in detail highlighting the different techniques used for the same. They have also emphasized on handwritten text recognition which is one of the complex fields. From their study it has been found that best results can be had with reduced computation time, and it is possible to segment multilingual characters and enhance the character recognition rate.

Sai Abhishikth Ayyadevara, P N V Sai Ram Teja, Rajesh Kumar M [9],this paper deals with two different proposals of machine learning techniques. The first one was a new feature extraction technique, including the feature of three different existing feature extraction techniques. While the second one includes the analysis of the performance of three different neural networks for two different feature techniques- geometric and gradient. After doing all the survey, they concluded that the Convolutional neural network is most efficiently absorbed through the Levenberg-Marquardt algorithm.

Kukich[10] suggested using a n-gram dictionary or method based on the errors and returning the possible word to the dictionary using mathematical steps. These methods may reduce the total number of OCR errors in standard language names, but it is possible that the words may be correctly identified that are not in the dictionary of geographical names.

## 2.3. Related Work

This subsection of the paper deals with the efforts carried out by various developers towards the core of recognizing text from different images.

Yang Zhang and Hao Zhang HaoranLi [11] described in their paper, an information extraction pipeline used for event flyers. The major steps in the pipeline include image capture and upload, image preprocessing, text detection, OCR and NLP information extraction. The paper lists several situations where a raw image could cause inaccurate results. The OCR engine would assume the picture is taken from a perpendicular upright view, but images taken from a handheld camera could contain distortions. The illumination of the image not being uniform throughout and an image containing multiple blocks of text of different sizes and colors could also affect the output. The image preprocessing methods included were edge detection, geometric correction(transformation), and Binarization.

Lavanya Bhaskar and R Ranjit [12] discuss an event planner for the brochure images, that implements text extraction by convolution followed by MSER feature extraction and Stroke width method. The event planner then directly links the event text to the google calendar for scheduling the events. However, the algorithm is not tested for event information taken from handwritten images and complex font text present in the images.

Brijesh Kumar Y. Panchal, and Gaurang Chauhan [13] proposed an implementation on the Android Application to extract using Tesseract OCR in which the following concepts are used, which are Adaptive Thresholding, Connected Component, Fine Lines, and Recognize Word. Using this Optical Character Recognition (OCR) Technology. The Application generates text, which is printed on a clean, B/W or colorful background and then can be converted into a computer readable form ASCII. With the help of this Android Application using Tesseract OCR, the system has two ways for Text Extraction. The first one is to capture a photo while the second one uploads an image from the gallery. After that the system can proceed as per the user requirement which portion of the image they want to crop or edit. After editing the picture, it converts into the text. This Android Application is for two languages, English and Hindi.

Salvador España-Boquera, Maria J. C. B., Jorge G. M., and Francisco Z. M. [14], this paper outlines the hybrid Hidden Markov Model (HMM) is used to conceive the unconstrained offline handwritten texts. The main characteristics of the recognition systems is to produce a new way in the form of preprocessing and recognition which are both based on ANNs. The preprocessing is used to clean the images and to enhance the non-uniform slant and slope correction. Whereas the recognition is used to estimate the emission probabilities.

K.Gaurav, Bhatia P. K. [15] , this paper deals with assorted pre-processing techniques used for handwritten recognition which consists of different images starting from a simple handwritten document and extending its radius to complex background and diverse image intensities. The pre-processing techniques that were included are contrast stretching, noise removal techniques, normalization, and segmentation, binarization, morphological processing techniques. They concluded that no technique for preprocessing can single handedly be used to produce an image. All the techniques go hand in hand. Even though after applying all the said techniques, the accuracy of the image is not up to the mark.

## 2.4. Tools/Technologies Required

The following technologies and tools are identified to develop our proposed solution-

A. **Tesseract**
Tesseract is an open-source text recognition (OCR) Engine, available under the Apache 2.0 license. It can be used directly, or (for programmers) using an API to extract printed text from images. It supports a wide variety of languages.

B. **Python**
Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

C. **Tkinter**
The tkinter package is the standard Python interface to the Tcl/Tk GUI toolkit. Tkinter is used in the project to build the interactive GUI application that acts as an abstraction of the driver code for the user.

D. **OpenCV**
OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.

E. **Pillow**
Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating, and saving images.

F. **NLTK**
NLTK (Natural Language Toolkit) Library is a suite that contains libraries and programs for statistical language processing. It is one of the most powerful NLP libraries, which contains packages to make machines understand human language and reply to it with an appropriate response.

# 3. DESIGN OF THE PROPOSED SYSTEM

The following chapter describes the high-level design of the system and the different steps and conditions that will be used to test the working of the system.

## 3.1. Block Diagram

As shown in figure 3.1, the user interacts with the system through an interface – here we have implemented a web browser. The user is directed to upload an image of the document that they'd like to extract text from. The driver code then uploads the image to the backend system where it undergoes a series of preprocessing steps, followed by text detection and extraction with the help of an OCR engine. The extracted text is in a raw data form, meaningful data is extracted from it with the help of a system of Natural Language Processing Techniques.

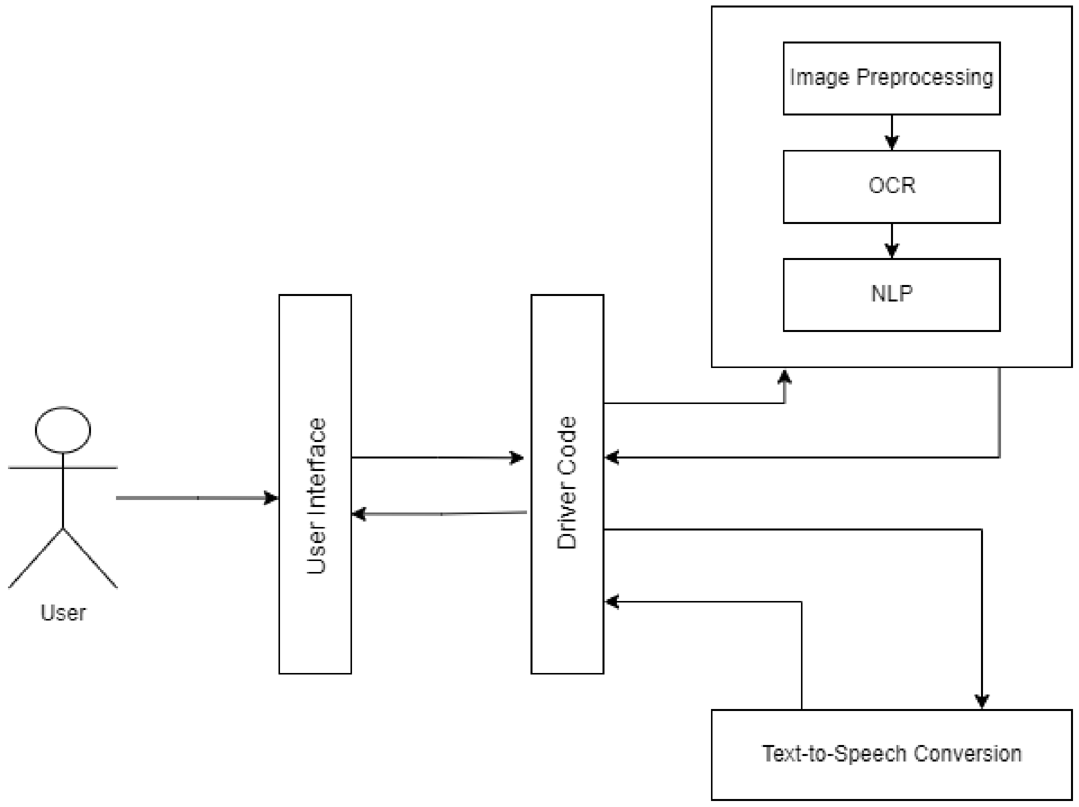


*Figure 3.1: Block Diagram of the System*

## 3.2 Module Description and Theoretical Foundation

The whole work of this dissertation is categorized into three major modules – Image Pre-processing, Text Extraction, and Post-processing modules.

## 3.2.1 Image Pre-processing

The Pre-processing module takes in an image and aims to eliminate the challenges that may occur while detecting text, caused due to noise, blurring effects, uneven lighting, skewness. In this stage, the image input scanned and uploaded by the user is processed to rotate to correct skew and detect and crop the document out of the background, remove noise, sharpen the image, and remove the blurriness that may affect the image during extraction. The preprocessed image is then sent through for text-extraction.

When an image of a receipt is taken, it is highly unlikely that all images are taken in high definition, and with the receipts containing no creases. These images might not just contain the receipt, but also the surrounding area the receipt was placed on or held against. This will decrease the chances of the text being recognized. So first we find a way for the edges of the receipt to be detected and cropped off the image. This is a task that must be handled by the software as the application is aimed towards decreasing the work of the user. With OpenCV, this task can be accomplished if proper steps are followed. The edges of the contours present in the image are detected and the document, here the receipt is singled out.

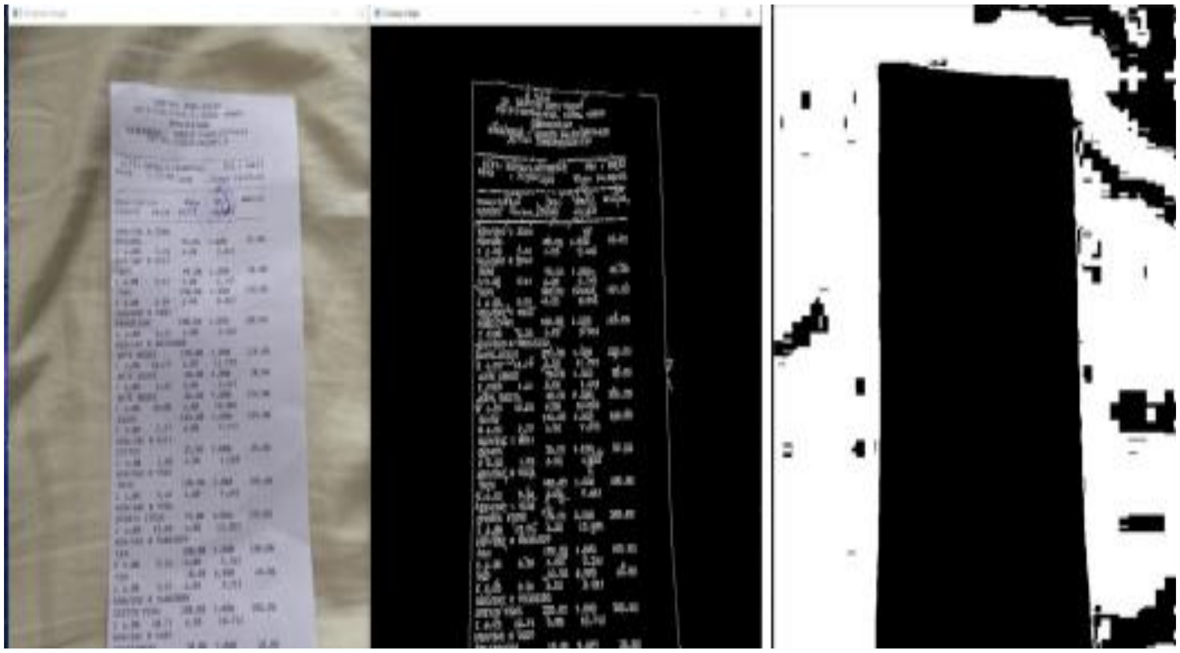
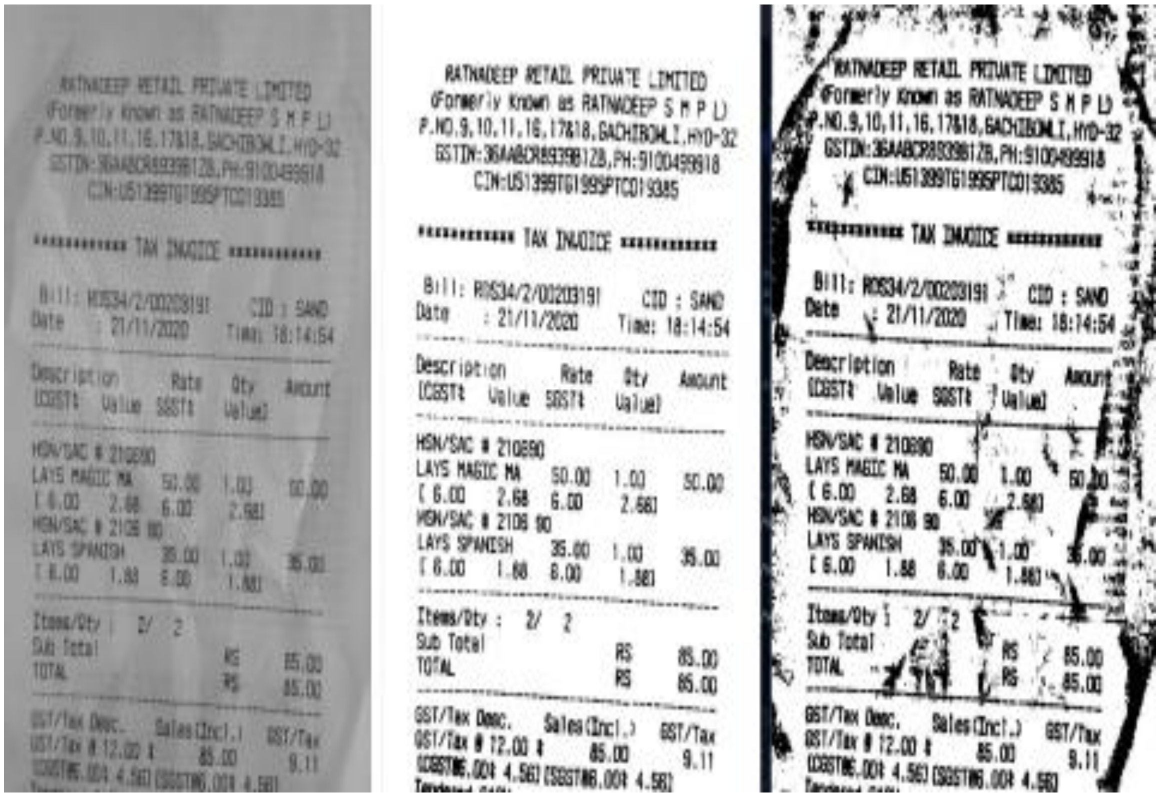


*Figure 3.1: Edge Detection*

Once the dimensions of the receipt are saved, they are sent as input to the preprocessing methods to enhance the quality of the image. For the text on the receipt to be recognized by the OCR engine, the images need to be preprocessed. Most OCR engines work well on Black & White images. Common preprocessing methods include – Gray scaling, Thresholding (Binarization) and Noise removal. Gray scaling is simply converting a RGB image to a grayscale image. Noise removal is done using morphology techniques such as Blur and Dilate.

Thresholding involves the assignment of pixel values in relation to a threshold value provided. Each pixel value is compared with the threshold value. If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value (generally 255). OpenCV provides various thresholding options - Simple Thresholding, Adaptive Thresholding



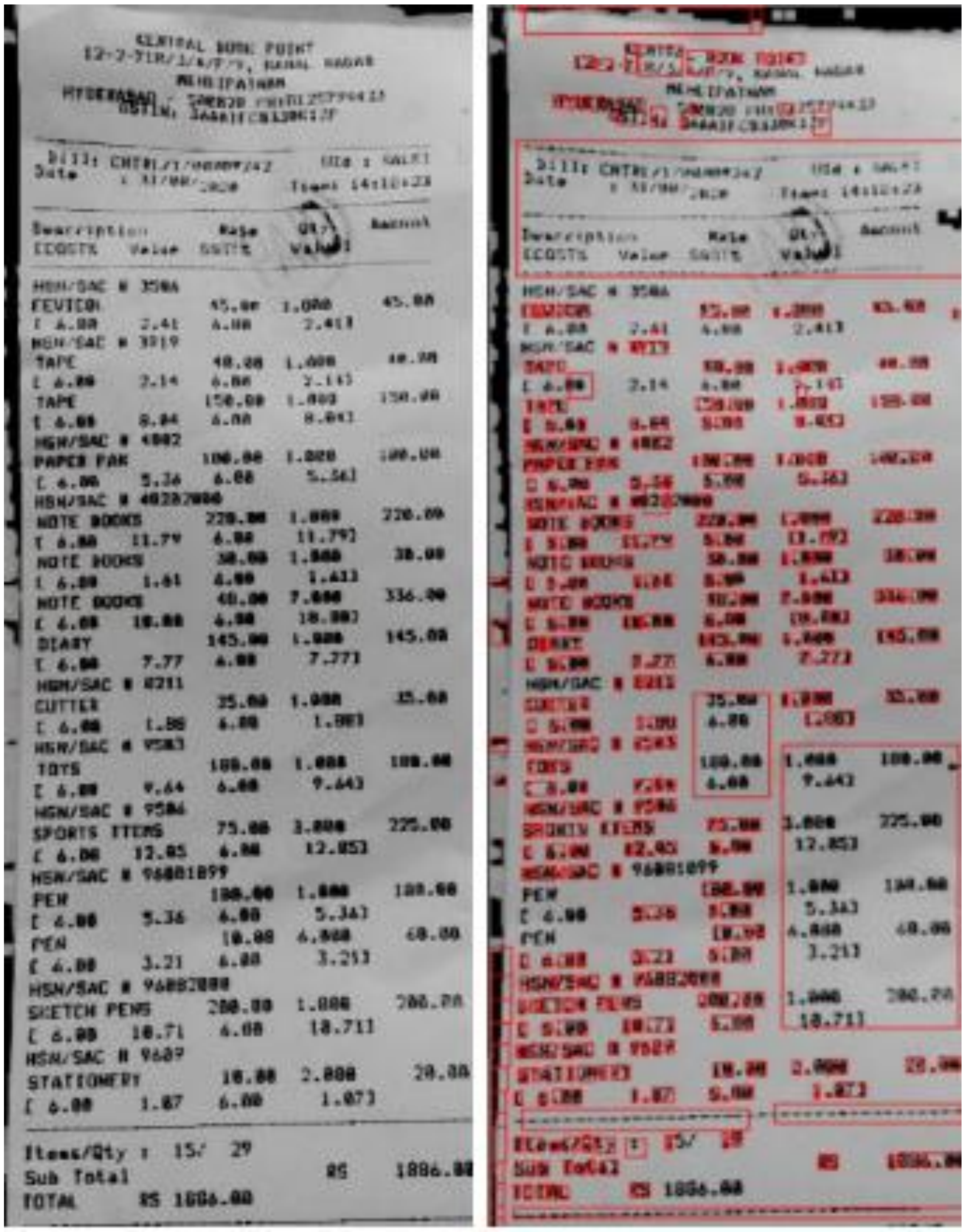
```
    RATNADEEP RETAIL PRIVATE LIMITED
    (Formerly Known as RATNADEEP S M P L)
  P.NO.9,10,11,16,17&18,GACHIBOWLI,HYD-32
    GSTIN:36AABCR8939B1ZB,PH:9100499818
        CIN:U51399TG1995PTC019385

*********** TAX INVOICE ***********

Bill: R0534/2/00203191      CID : SAND
Date    : 21/11/2020       Time: 18:14:54
------------------------------------------
Description       Rate    Qty     Amount
(CGST%   Value   SGST%   Value)
------------------------------------------
HSN/SAC # 210690
LAYS MAGIC MA      50.00   1.00     50.00
( 6.00    2.68    6.00    2.68)
HSN/SAC # 2106 90
LAYS SPANISH       35.00   1.00     35.00
( 6.00    1.88    6.00    1.88)
------------------------------------------
Items/Qty :   2/  2
Sub Total
TOTAL                       RS      85.00
                            RS      85.00
------------------------------------------
GST/Tax Desc.     Sales(Incl.)   GST/Tax
GST/Tax # 12.00 %      85.00        9.11
(CGST@6.00% 4.56) (SGST@6.00% 4.56)
```

Figure 2.3: Thresholding

Once the preprocessing phase is complete, we can expect a higher chance of our text being recognized correctly by the OCR engine

## 3.2.2 Text Extraction

The next step is Optical Character Recognition (OCR). It is a widespread technology used to recognize text inside images, such as scanned documents and photos. OCR technology is used to convert virtually any kind of images containing written text (typed, handwritten or printed) into machine-readable text data. Once a scanned paper document goes through OCR processing, the text of the document can be edited with word processors. Before OCR technology was available, the only option to digitize printed paper documents was to manually re-type the text. Not only was this very time consuming, but it also came with lots of typing errors. Fortunately, there are very good open-source OCR libraries. The most popular option was the Tesseract OCR engine. OCRopus was also a good option, especially for those who want to explore the inner workings of OCR. We used Tesseract-OCR, Google’s open-source OCR engine. By using a method to draw boxes around the text that has been detected by the OCR engine, as shown in Figure 3.4**,** we can observe that most of the text present on the receipt was detected. Here, we use pytesseract, which is a simple wrapper around Tesseract. Using pytesseract is simple: call *image_to_string()* to convert the image into a single formatted string. Calling *image_to_data()* will return individual text fragments with recognition confidence and other useful information.



```
              CENTRAL BOOK POINT
      12-2-718/1/A/F/9, KAMAL NAGAR
               MEHDIPATNAM
   HYDERABAD - 500028 PH:[illegible]
        GSTIN: 36AAIFC3130K1ZF

 Bill: CNTRL/1/[illegible]/42        Uid : SALE1
 Date     : 31/08/2020          Time: 14:10:23

 Description          Rate     Qty       Amount
 CGST%     Value     SGST%     Value

 HSN/SAC # 3506
 FEVICOL              45.00   1.000      45.00
 [ 6.00      2.41     6.00     2.411
 HSN/SAC # 3919
 TAPE                 40.00   1.000      40.00
 [ 6.00      2.14     6.00     2.143
 TAPE                150.00   1.000     150.00
 [ 6.00      8.04     6.00     8.041
 HSN/SAC # 4802
 PAPER PAK           100.00   1.000     100.00
 [ 6.00      5.36     6.00     5.361
 HSN/SAC # 48202000
 NOTE BOOKS          220.00   1.000     220.00
 [ 6.00     11.79     6.00    11.791
 NOTE BOOKS           30.00   1.000      30.00
 [ 6.00      1.61     6.00     1.611
 NOTE BOOKS           48.00   7.000     336.00
 [ 6.00     18.00     6.00    18.001
 DIARY               145.00   1.000     145.00
 [ 6.00      7.77     6.00     7.771
 HSN/SAC # 8211
 CUTTER               35.00   1.000      35.00
 [ 6.00      1.88     6.00     1.881
 HSN/SAC # 9503
 TOYS                180.00   1.000     180.00
 [ 6.00      9.64     6.00     9.643
 HSN/SAC # 9506
 SPORTS ITEMS         75.00   3.000     225.00
 [ 6.00     12.05     6.00    12.053
 HSN/SAC # 96081099
 PEN                 100.00   1.000     100.00
 [ 6.00      5.36     6.00     5.361
 PEN                  10.00   6.000      60.00
 [ 6.00      3.21     6.00     3.211
 HSN/SAC # 96082000
 SKETCH PENS         200.00   1.000     200.00
 [ 6.00     10.71     6.00    10.711
 HSN/SAC # 9609
 STATIONERY           10.00   2.000      20.00
 [ 6.00      1.07     6.00     1.071

 Items/Qty :  15/  29
 Sub Total                       RS    1886.00
 TOTAL       RS 1886.00
```

*Figure 3.3: Text Detection done by the OCR Engine*

### 3.2.3. Post-processing with Natural Language Processing

Natural Language Processing, or NLP for short, is defined as the automatic manipulation of natural language, like speech and text, by software. NLP primarily comprises of Natural Language Understanding (human to machine) and Natural Language Generation (machine to human). Our project deals with NLU. NLU aids in extracting valuable information from text such as social media data, customer surveys, and complaints. In our project we used a common technique of NLP called Named Entity Recognition (NER). This is the most basic and useful technique in natural language processing for extracting entities in text. 4 Named entity recognition (NER) identifies entities such as people, locations, organizations, dates, etc. from the text. For example, for text obtained from our receipt image, the following output can be expected: Vendor Name, Date, Total, Address. NER is generally based on grammar rules and supervised models. However, there are NER platforms such as open NLP that have pre-trained and built-in NER models. For the given entity categories above, a JSON file is created containing the extracted classes. In many cases, especially for data other than Total Amount, labels aren't found for the rest. In such a case, the model requires us to create labels for each of the inputting tokens (words or characters), which needs to be created separately. For this, we train a model with a predefined dataset containing images of receipts and their corresponding data in a JSON file. The model learns to identify what an address is, what a vendor name typically etc. In our model, we were able to successfully train it using a dataset found on Kaggle, to identify vendor name and Total amount accurately

### 3.2.4. Integrating into an Application

After the code to parse the receipt image is developed, the entire pipeline needs to be deployed. This was done using a Python framework called Django. Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. The web application was also built using Django. Django takes care of much of the hassle of web development, so that the developer can focus on writing the application without needing to spend time on building it from scratch. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support. The web application consisted of a single home website, which navigates the user to upload the image, as well as download the output text file and audio file to their local system.

A system Graphical User Interface (GUI) was also implemented to be used as an in-house application for the university. The GUI application was built with the help of Python's GUI toolkit, tkinter.

# 4. IMPLEMENTATION OF THE PROPOSED SYSTEM

## 4.1. Architecture Diagrams

The following sub-sections describe the architectural design, entities and data flow of the system with the help of flowcharts, data flow diagrams, entity relationship diagrams and use case diagrams.

### 4.1.1. Flowchart

Figure 4.1 shows the flow from one sub process to another in our proposed pipeline. When the user uploads an image of their document to the interface, it goes through a series of preprocessing methods to enhance the text present on the image and the text is then detected and extracted. Useful information is then extracted from this raw text to form structured data as well as in audio format.

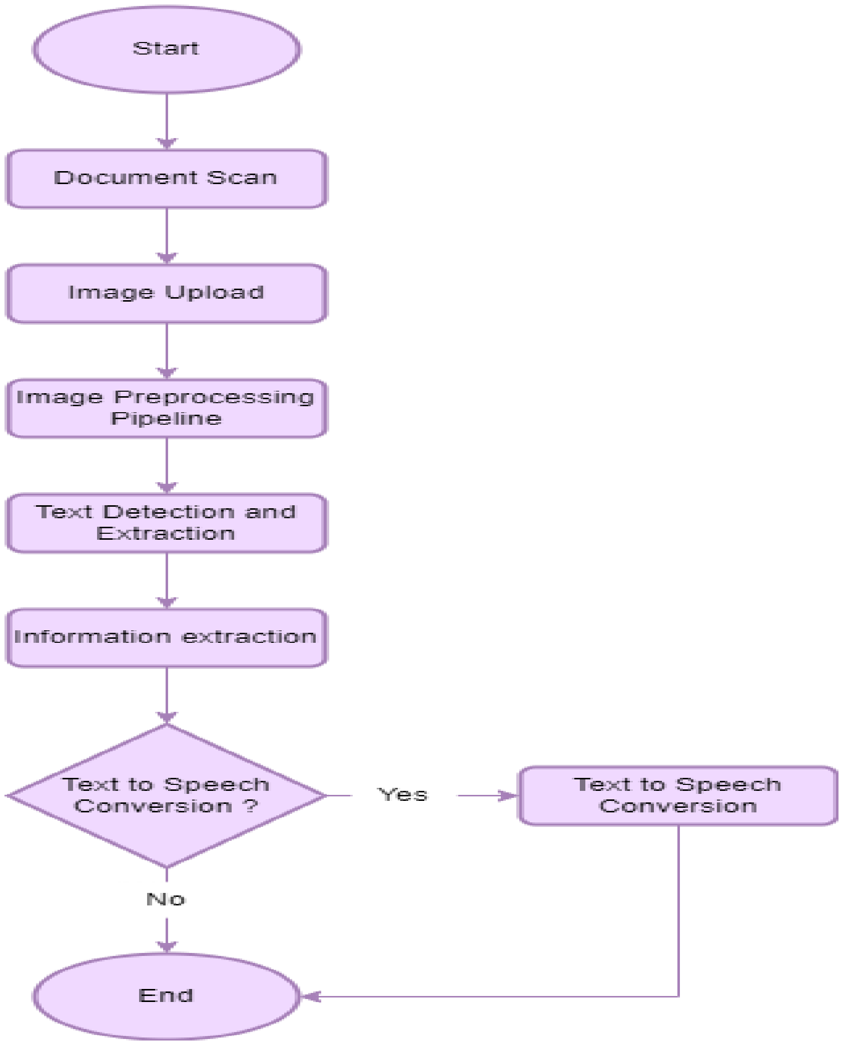

Start
Document Scan
Image Upload
Image Preprocessing Pipeline
Text Detection and Extraction
Information extraction
Text to Speech Conversion ?
Yes
Text to Speech Conversion
No
End

*Figure 4.1: Flowchart depicting adopted methodology*

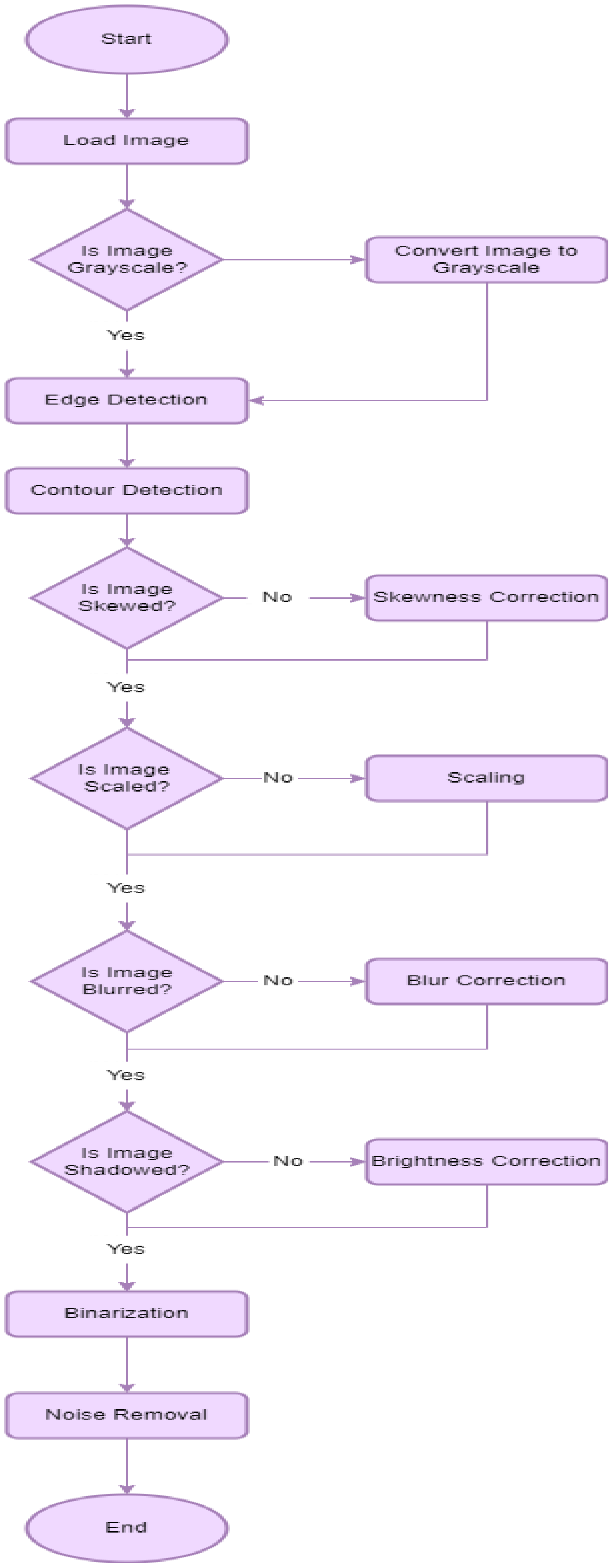


*Figure4.2: Image preprocessing pipeline*

Figure 4.2 describes the various subprocesses involved in the image preprocessing pipeline. The image is converted to grayscale and then corrected in case the image has a gradient brightness, is at an angle, contains a background, is blurry or contains noise.

### 4.1.2. Data Flow Diagram

The Data Flow Diagram (DFD) depicts the logic models and expresses data transformation in a system. It includes a mechanism to model the data flow and supports decomposition to illustrate details of the data flows and functions.

**DFD Level 0**

This is the fundamental system model that represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows. The system is then decomposed and described as a DFD with multiple bubbles. Figure 7 depicts how the user interacts with the interface to upload an image of the document and retrieve structured data and text-to-speech converted data.

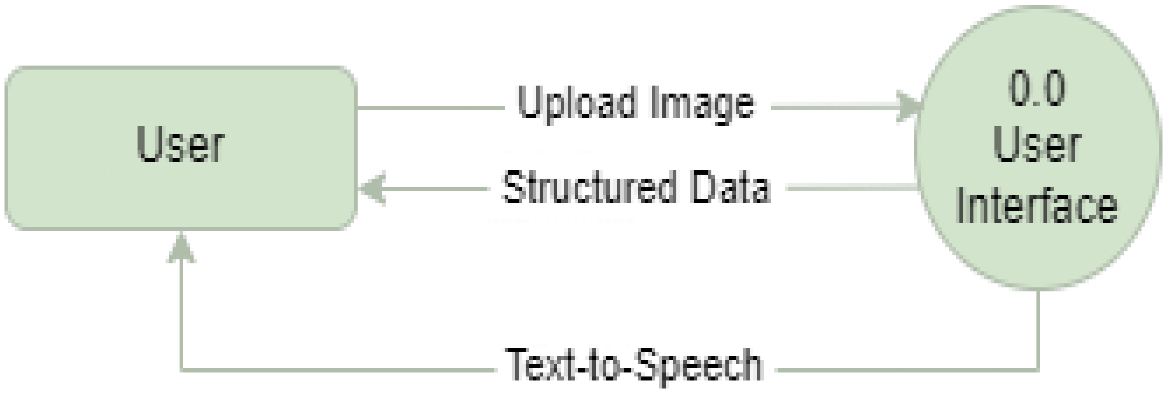


*Figure 4.3: Data Flow Diagram Level 0*

**DFD Level 1**

In Level 1 DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, the main objectives of the system are broken down to high level process of Level 0 DFD into subprocesses. It also projects or records the specific/necessary detail about the system’s functioning.

Figure 4.4 shows how a student login into the student, and how the web camera is enabled, and the student pictures are captured and then the image acquisition, detection, preprocessing and recognition is done with the help of student database. The diagram also illustrates the activities or functions, or modules managed by faculty and admin.

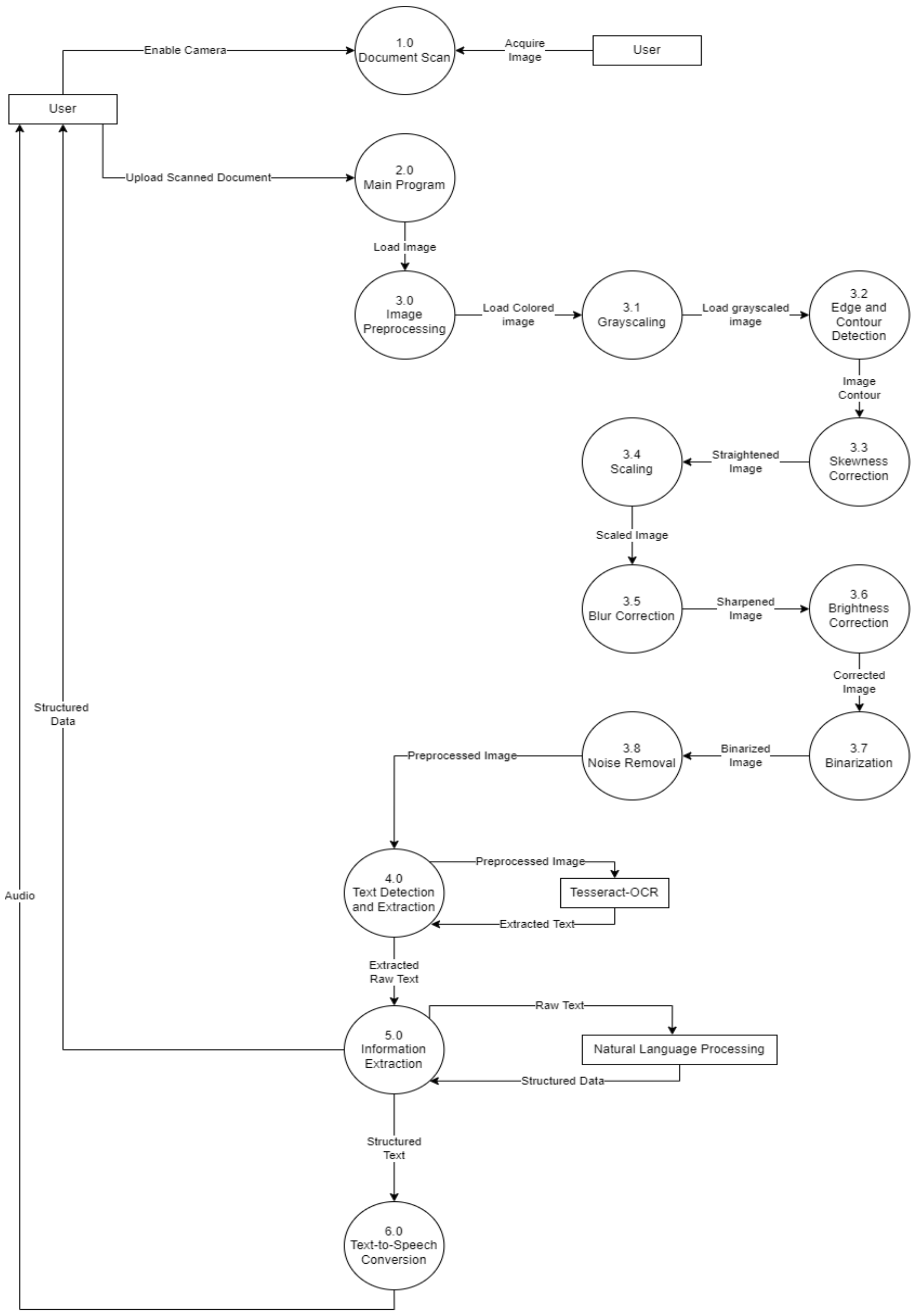


*Figure 4.4: Data Flow Diagram Level 1*

## 4.1.3. Entity – Relationship

An Entity Relationship Diagram (ERD) is a visual representation of different entities within a system and how they relate to each other. ER diagrams help to explain the logical structure of databases. They are created based on three basic concepts: entities, attributes, and relationships. It is used to represent the entity framework infrastructure. Figure 4.5 describes the entities and relationships present within the system.

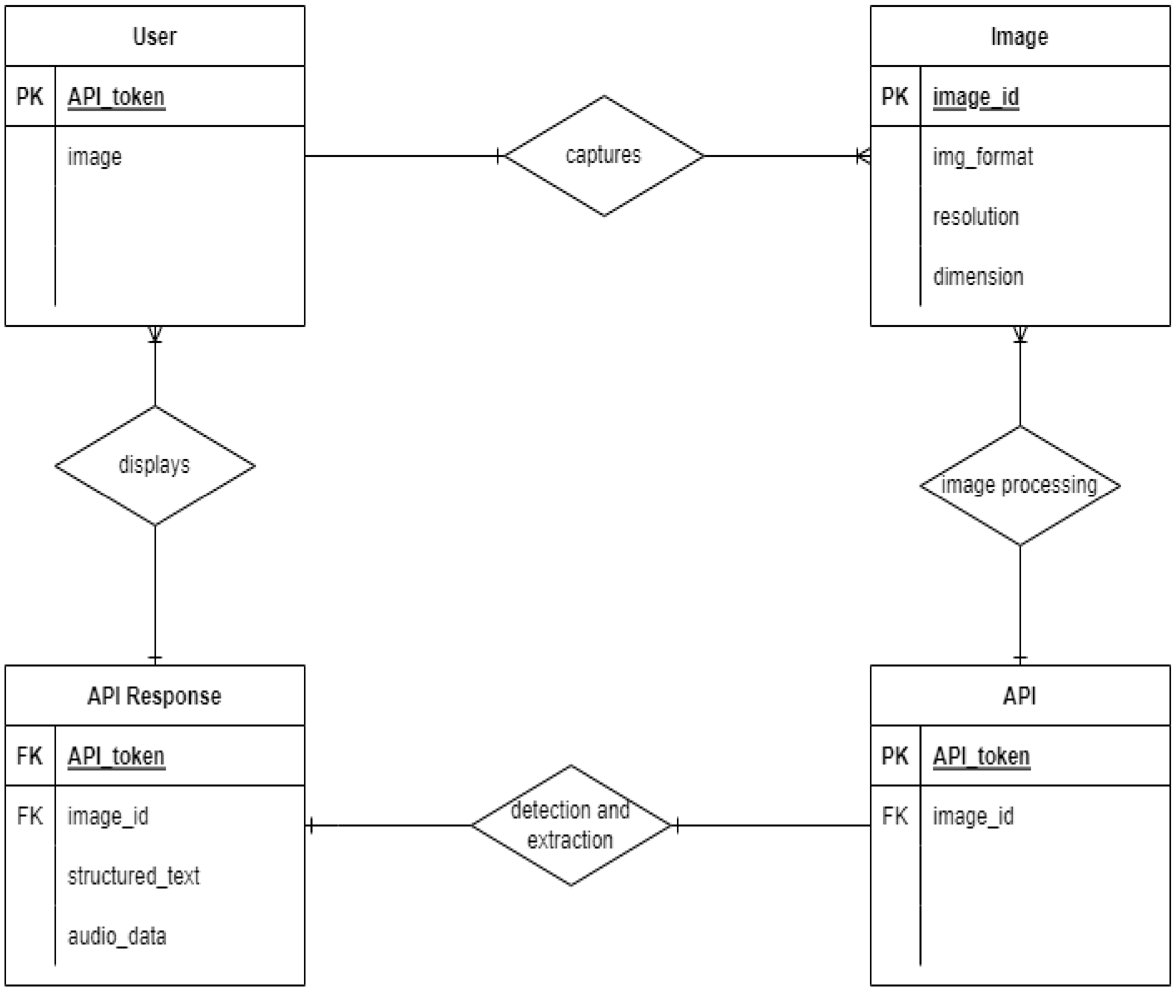


*Figure 4.5: Entity-Relationship Diagram*

## 4.1.4. Use Case Diagram

Use case diagram is dynamic in nature and there are some internal or external factors for making the interactions. These internal and external agents are known as actors. Use case diagrams consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. Hence to model the entire system, several use case diagrams are used.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. When a system is analyzed to gather functionalities, use cases are prepared and actors are identified. The use case diagrams are then modelled to present the outside view.

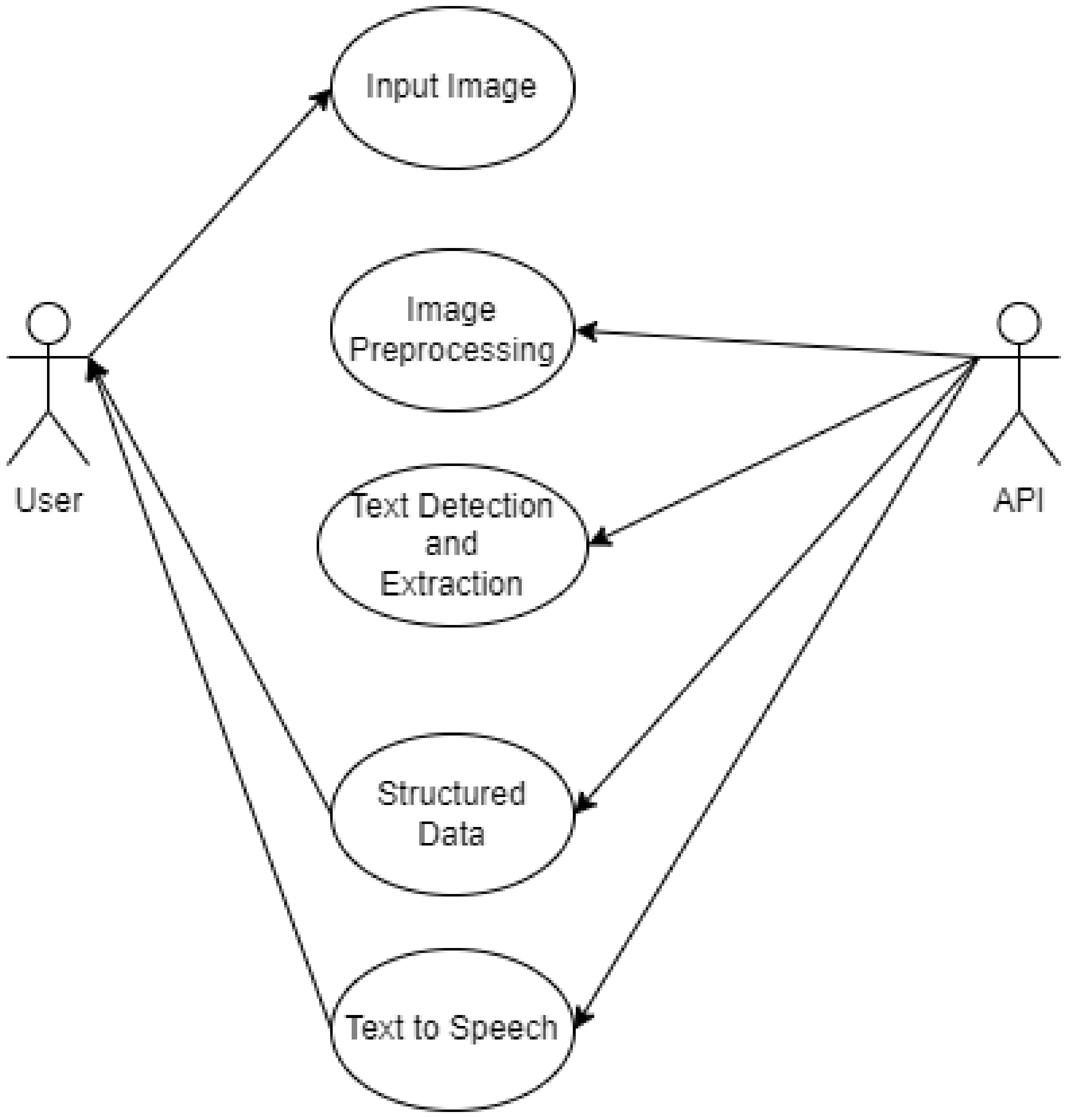


*Figure 4.6: Use Case Diagram*

### 4.1.5. Sequence Diagram

The diagram depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.

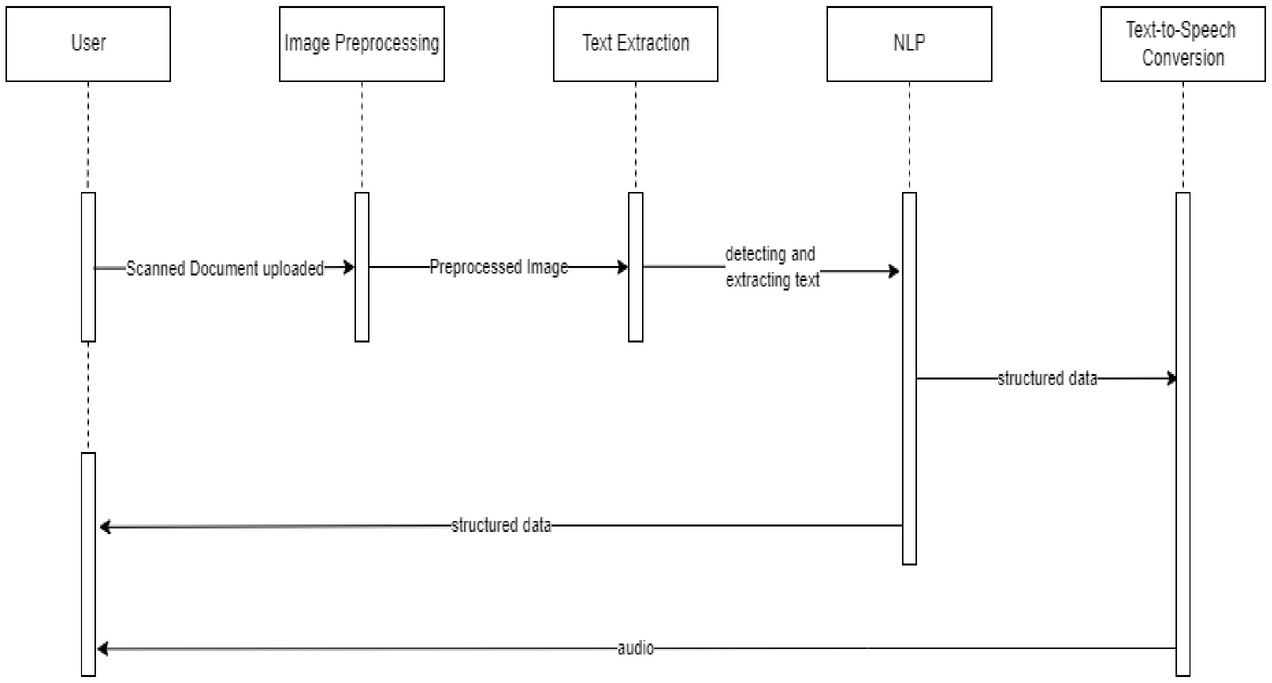


*Figure 4.7: Sequence Diagram*

## 4.2. Pseudocode

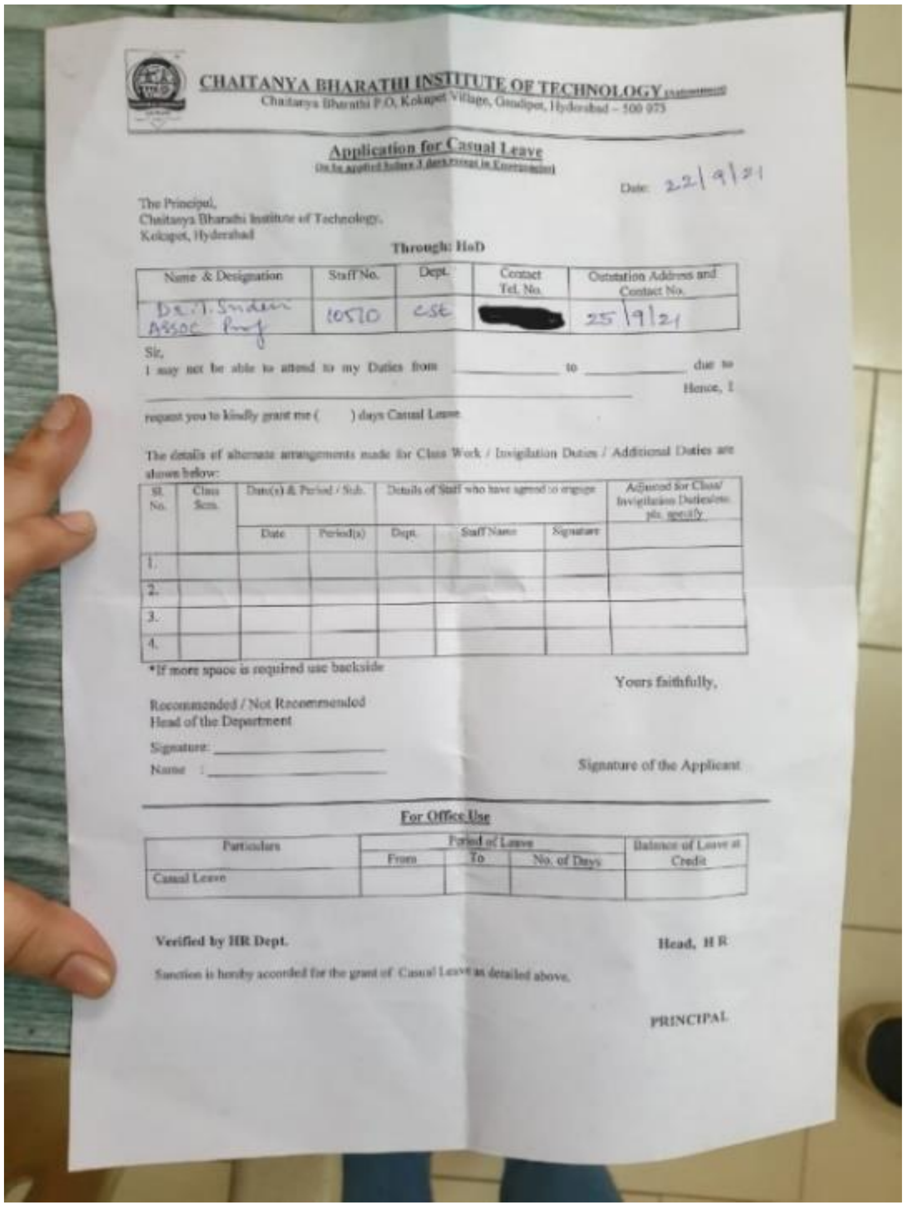
```
1. Function grayscale(image):
   1.1. if image is not a grayscale image, then convert image to grayscale
   1.2. return image
2. Function edge_detection(image):
   2.1. Define kernel size
   2.2. Dilate image
   2.3. Detect edges with the help of canny edge detection
   2.4. return edges
3. Function binarization(image):
   3.1. Binarize the image by adjusting a proper threshold
   3.2. Return image
4. Function contour_detection(image):
   4.1. Ensure image is in grayscale
```

```
    4.2. Ensure image has been binarized
    4.3. Apply Morphology
    4.4. Detect Contours using findContours(), with CHAIN APPROX method
    4.5. Calculate the area of various contours, the largest area found is the document
    4.6. Crop image
    4.7. Return image
5. Function skew_correction(image):
    5.1. Find the edges of the image
    5.2. Perform classic straight-line Hough transform between 0.1 - 180 degrees.
    5.3. Find line peaks and angles
    5.4. Round the angles to 2 decimal places and find the most common angle.
    5.5. Convert the angle to degree for rotation.
    5.6. Rotate image through angle
    5.7. Return image
6. Function blur_correction(image):
    6.1. Define sharpen kernel
    6.2. Sharpen image
7. Function shadow(image):
    7.1. Dilate and blur a copy of the image
    7.2. Check the difference between the image and the blurred result
    7.3. Normalize the Image
    7.4. Return Image
```

## 4.3. Dataset Description

In order to test the accuracy of the system, several different test cases are required in order to absolutely verify that each sub process is working efficiently.

The dataset used for the model comprises of a mix of real time images of documents collected by the authors. Figure 4.8 shows an employee leave application used by the teachers at our institute to apply for a leave. This image won’t just be helpful to test the accuracy of text extracted but can also be used to test whether the preprocessing module is able detect the document from its backgrounds and whether it is able to brighten the shadowed and wrinkled parts of the image.



CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (Autonomous)
Chaitanya Bharathi P.O, Kokapet Village, Gandipet, Hyderabad – 500 075

**Application for Casual Leave**
(to be applied before 3 days except in Emergencies)

Date: 22/9/21

The Principal,
Chaitanya Bharathi Institute of Technology,
Kokapet, Hyderabad

**Through: HoD**

| Name & Designation | Staff No. | Dept. | Contact Tel. No. | Outstation Address and Contact No. |
|---|---|---|---|---|
| Dr. T. [illegible] Assoc Prof | 10570 | CSE | [redacted] | 25/9/21 |

Sir,
I may not be able to attend to my Duties from __________ to __________ due to ________________________ Hence, I request you to kindly grant me (      ) days Casual Leave.

The details of alternate arrangements made for Class Work / Invigilation Duties / Additional Duties are shown below:

| Sl. No. | Class Sem. | Date(s) & Period / Sub. | | Details of Staff who have agreed to engage | | | Adjusted for Class/ Invigilation Duties/etc. pls. specify |
|---|---|---|---|---|---|---|---|
| | | Date | Period(s) | Dept. | Staff Name | Signature | |
| 1. | | | | | | | |
| 2. | | | | | | | |
| 3. | | | | | | | |
| 4. | | | | | | | |

*If more space is required use backside

Yours faithfully,

Recommended / Not Recommended
Head of the Department

Signature: ________________
Name : ________________

Signature of the Applicant

**For Office Use**

| Particulars | Period of Leave | | | Balance of Leave at Credit |
|---|---|---|---|---|
| | From | To | No. of Days | |
| Casual Leave | | | | |

**Verified by HR Dept.** **Head, H R**

Sanction is hereby accorded for the grant of Casual Leave as detailed above.

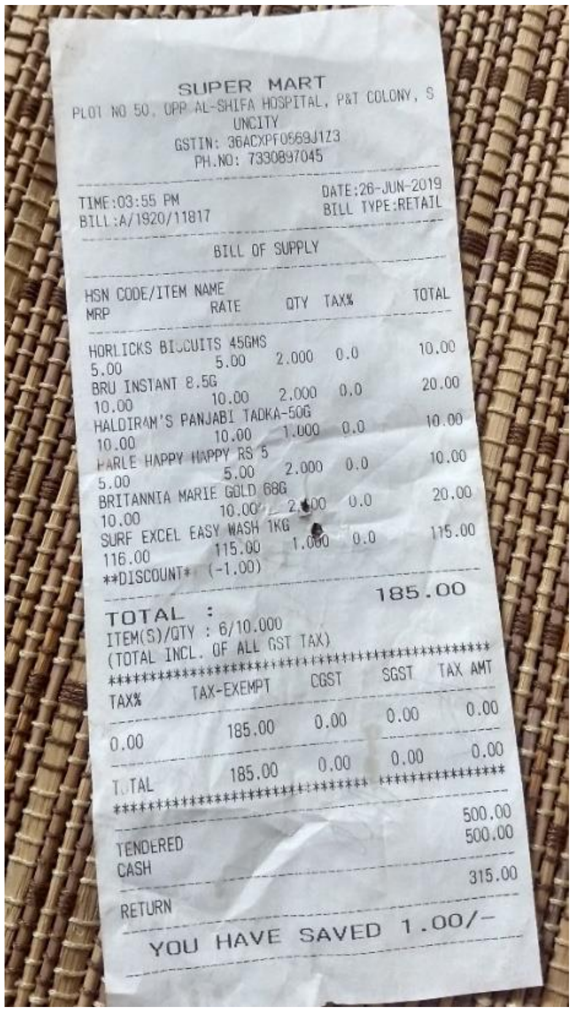
**PRINCIPAL**

*Figure 4.8: Test Input Image; Professor Leave Application form*

Images considered in the dataset for testing the module contain anomalies that we have targeted to fix.

Figure 4.9 is another example of a suitable image that we have used to test the system. It is at a slight angle, such images are difficult for the OCR Engine to detect text in. We aim at correcting the angle of such images and cropping the document from its background.

The image also contains wrinkles, and some parts of the text is slightly faint, we want to sharpen the text and smoothen the wrinkles.



```
                 SUPER  MART
PLOT NO 50, UPP AL-SHIFA HOSPITAL, P&T COLONY, S
                   UNCITY
            GSTIN: 36ACXPF0569J1Z3
              PH.NO: 7330897045
------------------------------------------------
                              DATE:26-JUN-2019
TIME:03:55 PM                 BILL TYPE:RETAIL
BILL:A/1920/11817
------------------------------------------------
                BILL OF SUPPLY
------------------------------------------------
HSN CODE/ITEM NAME                         TOTAL
MRP             RATE      QTY   TAX%
------------------------------------------------
HORLICKS BISCUITS 45GMS
5.00            5.00     2.000   0.0       10.00
BRU INSTANT 8.5G
10.00          10.00     2.000   0.0       20.00
HALDIRAM'S PANJABI TADKA-50G
10.00          10.00     1.000   0.0       10.00
PARLE HAPPY HAPPY RS 5
5.00            5.00     2.000   0.0       10.00
BRITANNIA MARIE GOLD 68G
10.00          10.00     2.000   0.0       20.00
SURF EXCEL EASY WASH 1KG
116.00        115.00     1.000   0.0      115.00
**DISCOUNT*   (-1.00)
------------------------------------------------
                                       185.00
TOTAL  :
ITEM(S)/QTY : 6/10.000
(TOTAL INCL. OF ALL GST TAX)
************************************************
TAX%     TAX-EXEMPT     CGST     SGST    TAX AMT
------------------------------------------------
0.00       185.00       0.00     0.00     0.00
------------------------------------------------
TOTAL      185.00       0.00     0.00     0.00
************************************************
------------------------------------------------
                                          500.00
TENDERED                                  500.00
CASH
------------------------------------------------
                                          315.00
RETURN
------------------------------------------------
      YOU  HAVE  SAVED  1.00/-
------------------------------------------------
```

*Figure 4.9: Test Input Image: Skewed capture of a receipt with a textured background*

# 5. RESULT

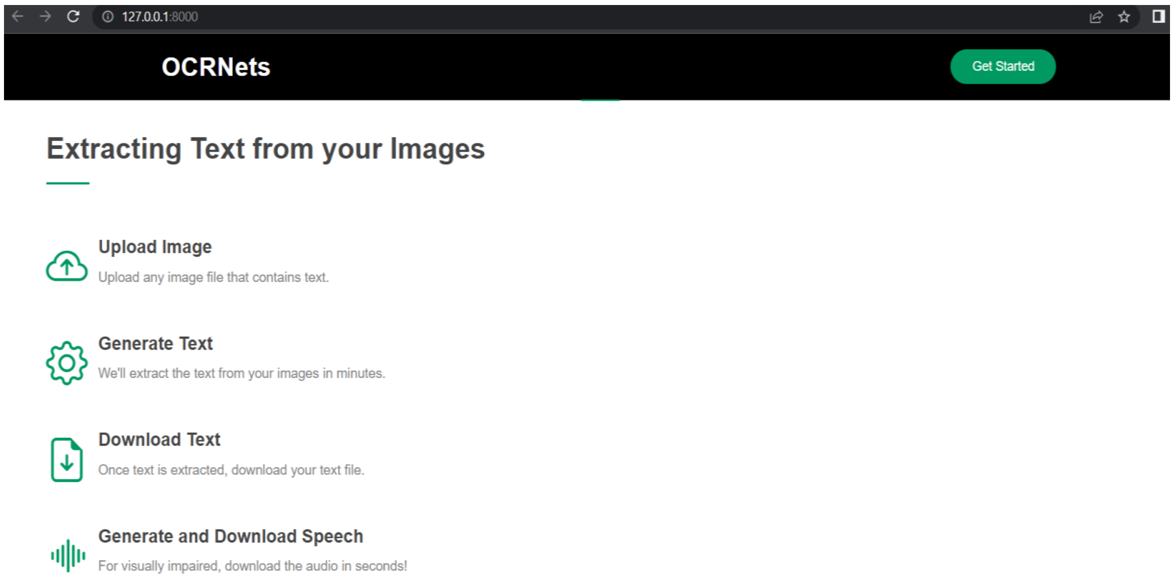
This chapter discusses the result of the text extraction from images. The software is tested using images of varied qualities and on a computer with the specification as follows, a processor of Intel Core i5-8250U CPU 1.60GHz 1.80GHz and RAM of minimum 4GB.

## 5.1. Results

This section will show the results from the two interfaces that we have built for this image preprocessing to information extraction pipeline. The first subsection displays images of the web interface, and the second subsection illustrates a system executable Graphical User Interface.

### 5.1.1. Web Application

Figures 5.1 – 5.4 illustrate the web interface of the system. Figure 5.1 is a screenshot of the first section of the web page which briefly summarizes the steps involved for extracting the text from the images uploaded by the user.



Extracting Text from your Images

Upload Image
Upload any image file that contains text.

Generate Text
We'll extract the text from your images in minutes.

Download Text
Once text is extracted, download your text file.

Generate and Download Speech
For visually impaired, download the audio in seconds!

*Figure 5.1: Website Interface, displaying a short overview of the process*

Figure 5.2 is the subsection of the web page where the user uploads the image.

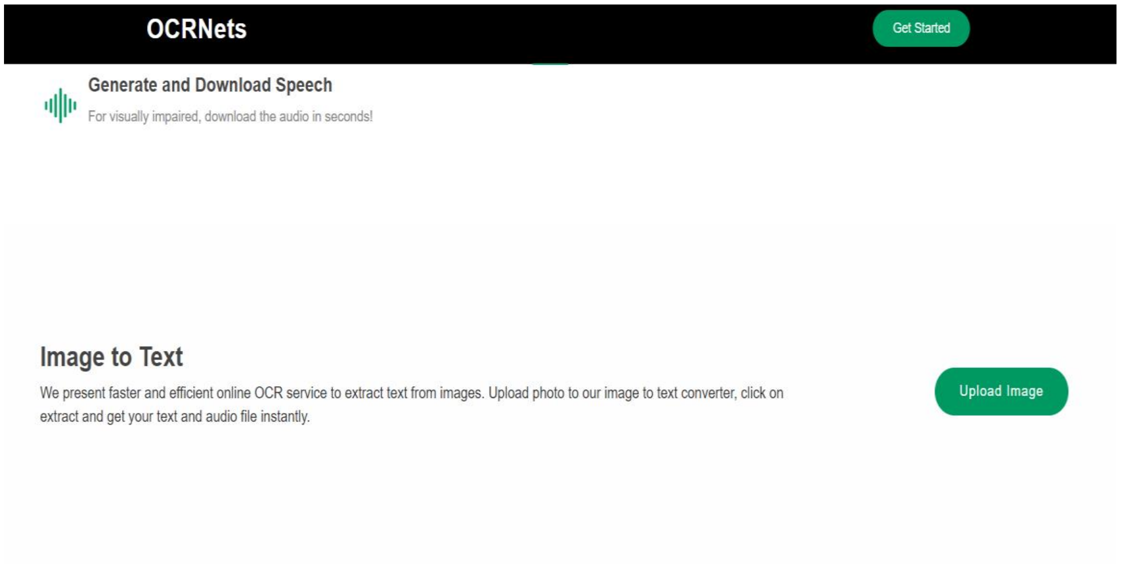

OCRNets
Get Started
Generate and Download Speech
For visually impaired, download the audio in seconds!
Image to Text
We present faster and efficient online OCR service to extract text from images. Upload photo to our image to text converter, click on extract and get your text and audio file instantly.
Upload Image

*Figure 4.2: Section on the website to upload image*

Upon clicking the upload image button, they are prompted to choose a file as depicted in figure 5.3.

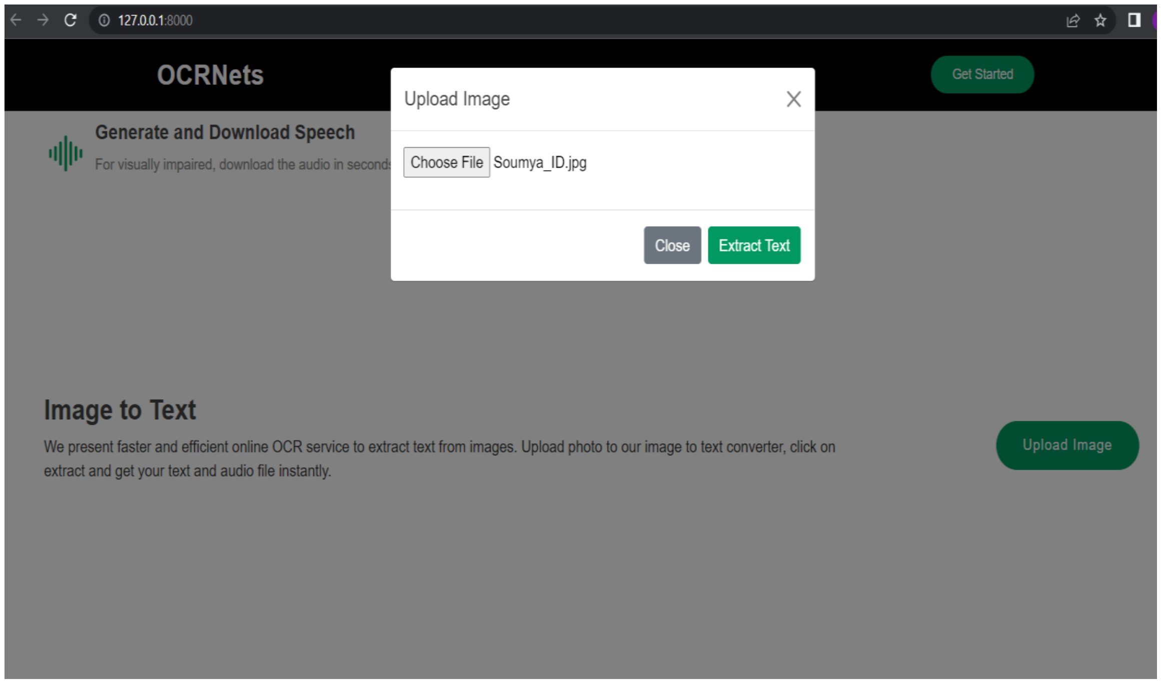
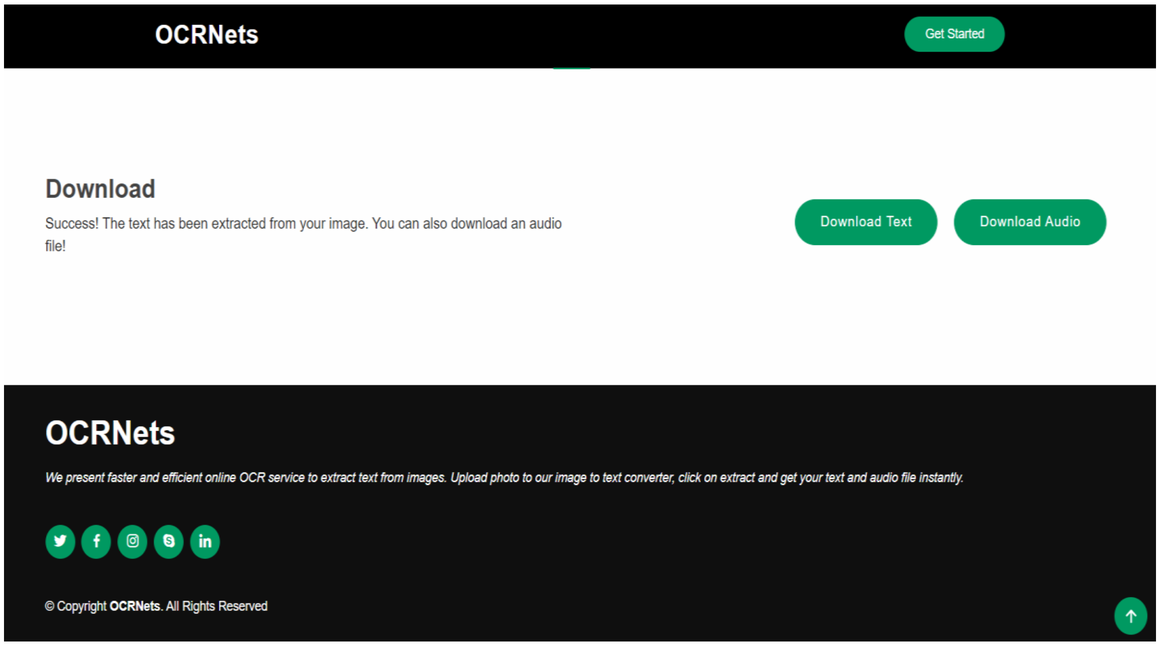

127.0.0.1:8000
OCRNets
Get Started
Generate and Download Speech
For visually impaired, download the audio in seconds
Upload Image
Choose File
Soumya_ID.jpg
Close
Extract Text
Image to Text
We present faster and efficient online OCR service to extract text from images. Upload photo to our image to text converter, click on extract and get your text and audio file instantly.
Upload Image

*Figure 5.3: Upload Image prompt; 'Choose File' directs the user to their file explorer*

Once the structured data is successfully extracted, text and audio files are generated for the user to download as shown in figure 5.4.



Download

Success! The text has been extracted from your image. You can also download an audio file!

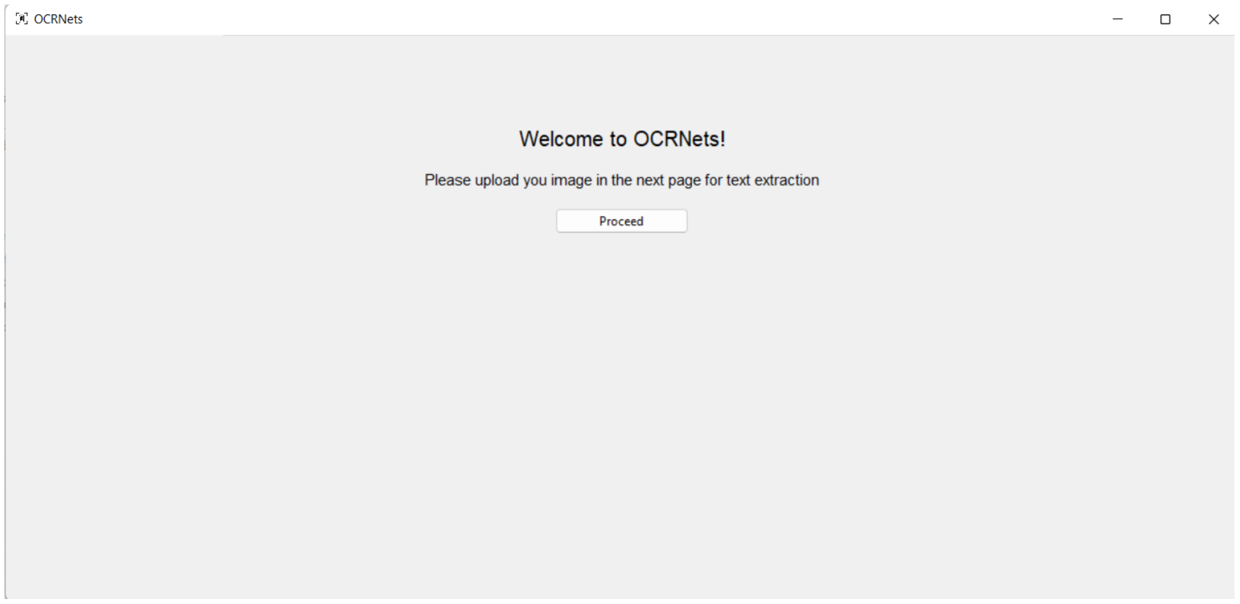
We present faster and efficient online OCR service to extract text from images. Upload photo to our image to text converter, click on extract and get your text and audio file instantly.

© Copyright **OCRNets**. All Rights Reserved

*Figure 5.4: Section on the website to download the data captured in the image*

## 5.1.2. Graphical User Interface

The Graphical User Interface (GUI) built to use as an inhouse application for our university is described in figures 5.5 – 5.10.

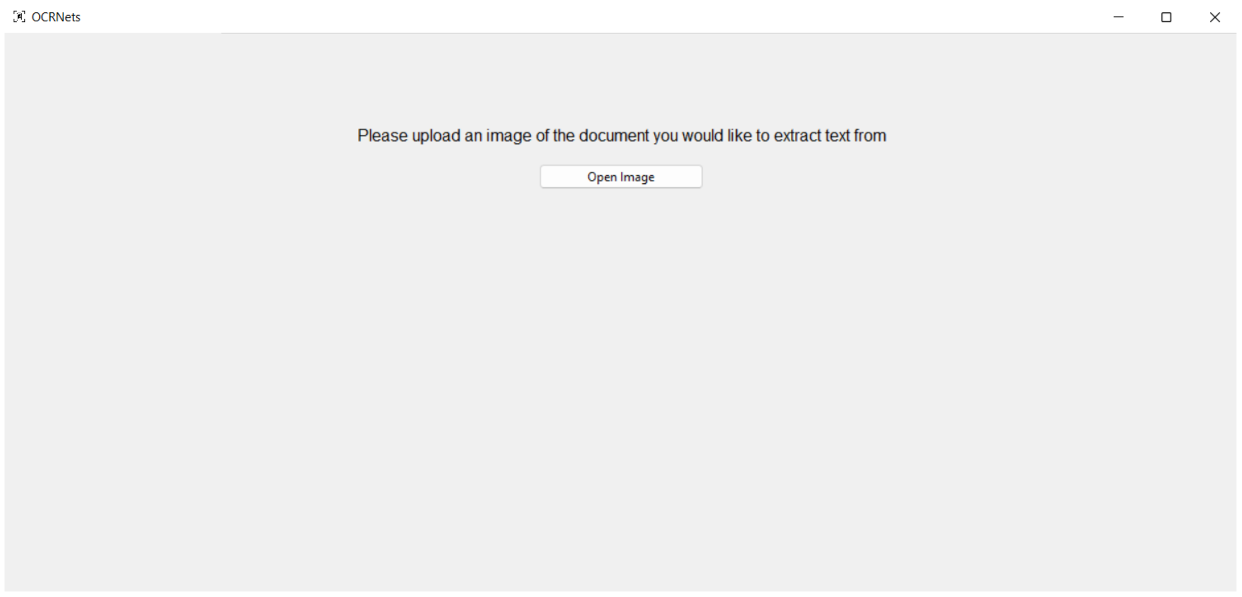


Welcome to OCRNets!

Please upload you image in the next page for text extraction

*Figure 5.5: Welcome Page of the Application*

Figure 5.5 is the page displayed to the user upon running the application. When they click to proceed to the next page, they are prompted to upload an image of their document as shown in figure 5.6.



Please upload an image of the document you would like to extract text from

*Figure 5.6: Upload Image prompt*

The Open Image button from figure 5.6, launches the file explorer allowing the users to browse for the image they require to extract text from as shown in Figure 5.6.

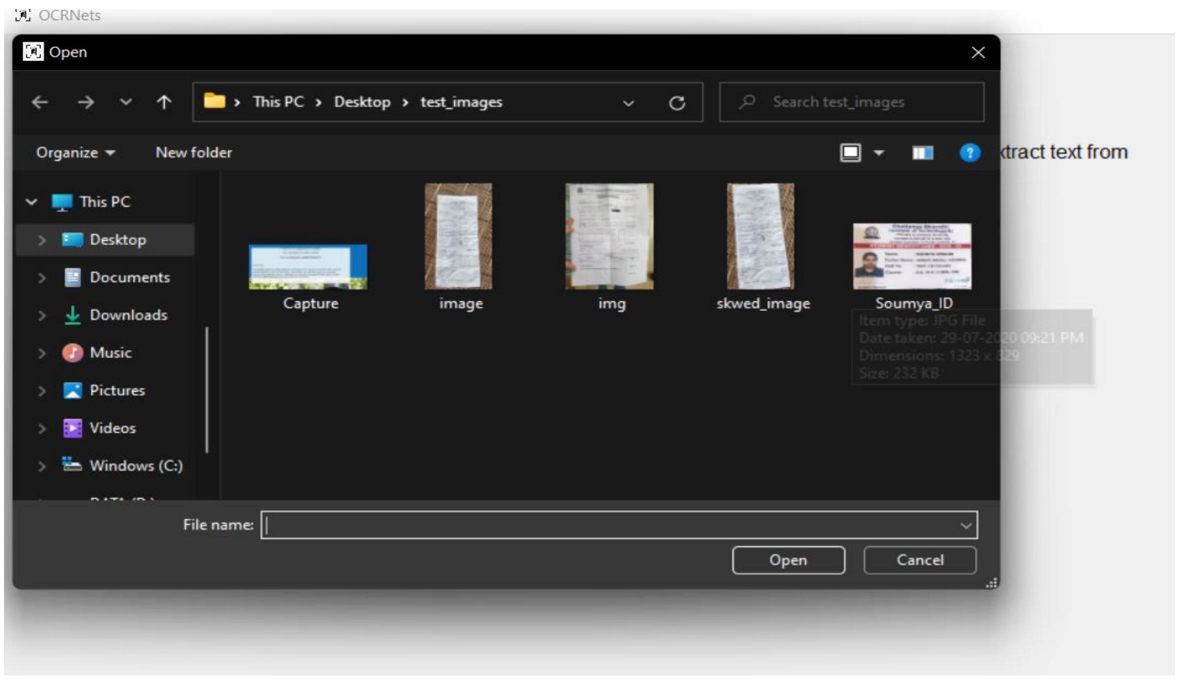
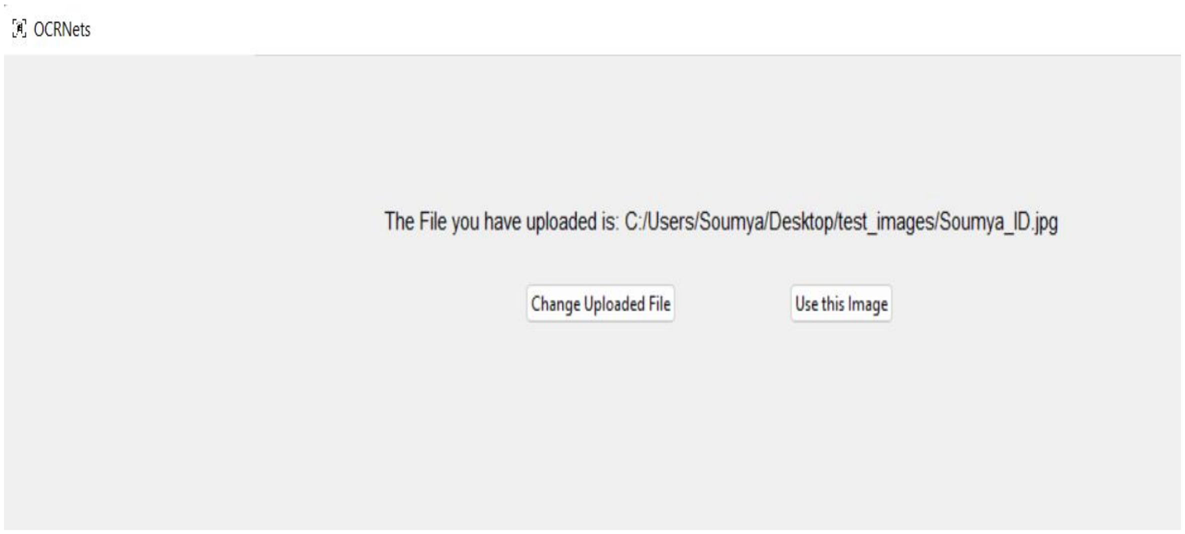


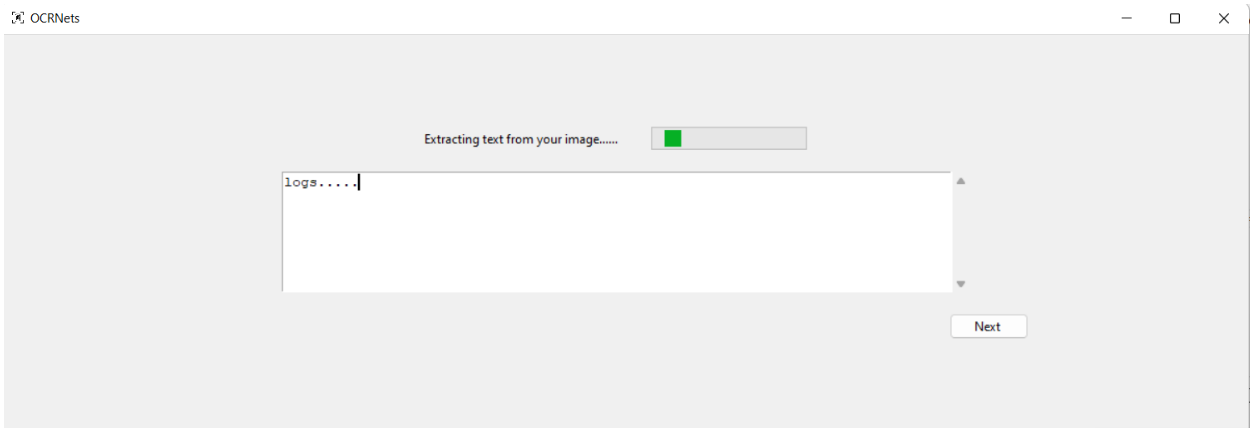
*Figure 5.7: File Explorer pop-up upon clicking "Upload Image"*



The File you have uploaded is: C:/Users/Soumya/Desktop/test_images/Soumya_ID.jpg

*Figure 5.8: Confirmation of Image Uploaded*

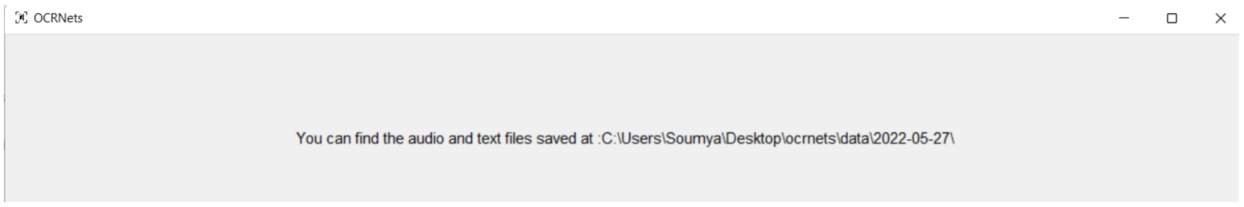
Once the file is uploaded, they are asked to confirm if they want to proceed with the uploaded image as shown in figure 5.8.



```
logs.....
```

*Figure 5.9: A dynamic progress bar is displayed, while waiting for the image preprocessing and text extraction process to complete*

While the image is being preprocessed, and the information is being extracted from the raw text, the user is provided with the progress bar to keep track of where the process stands along with a display of the logs generated for any interested developers as shown in figure 5.9. Once the process is done, they are immediately redirected to the location where the text and audio files are saved as shown in figure 5.10.

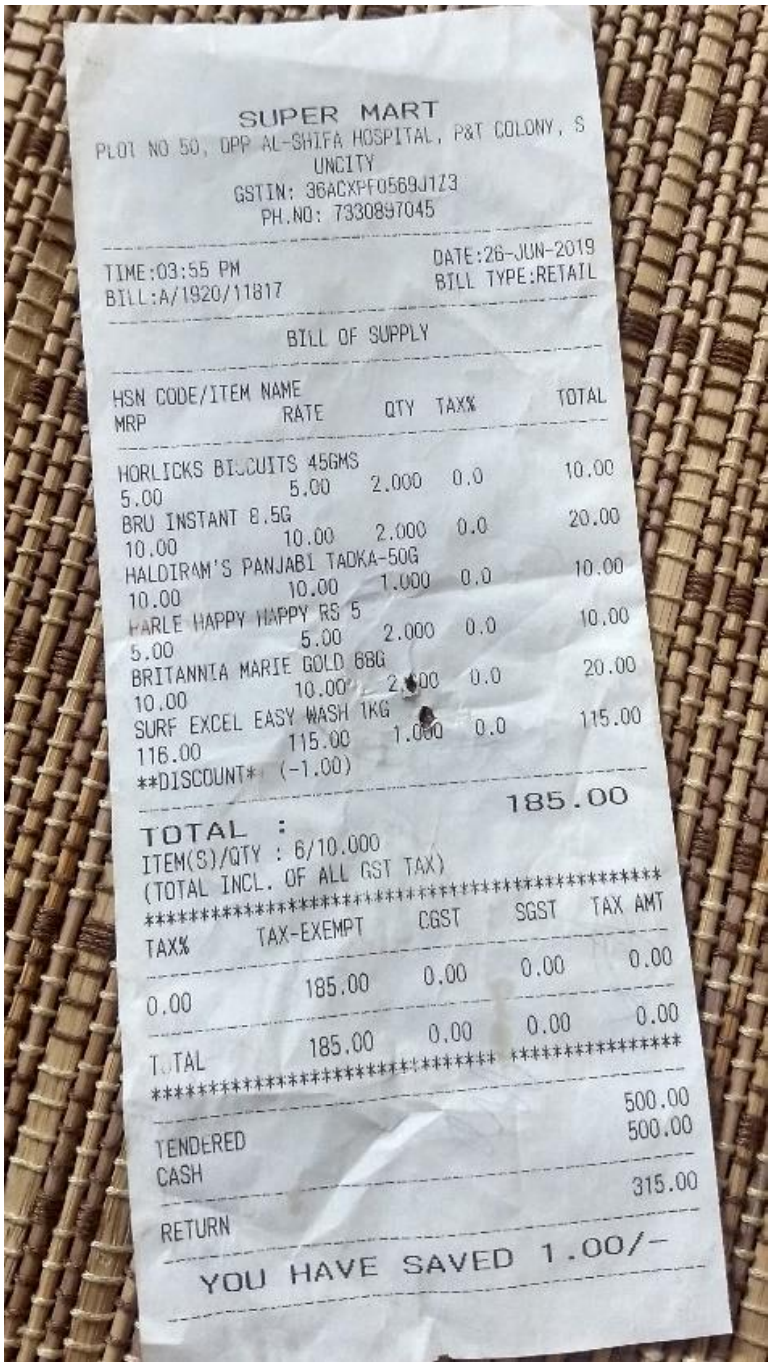


You can find the audio and text files saved at :C:\Users\Soumya\Desktop\ocrnets\data\2022-05-27\

*Figure 5: The Data File and Audio File are stored locally, and the user is directed to their location*

## 5.2. Output

This section will first describe the process with the help of a sample input image, describing the step-by-step process of going through the image preprocessing pipeline, and then displaying the text output for this sample input image.



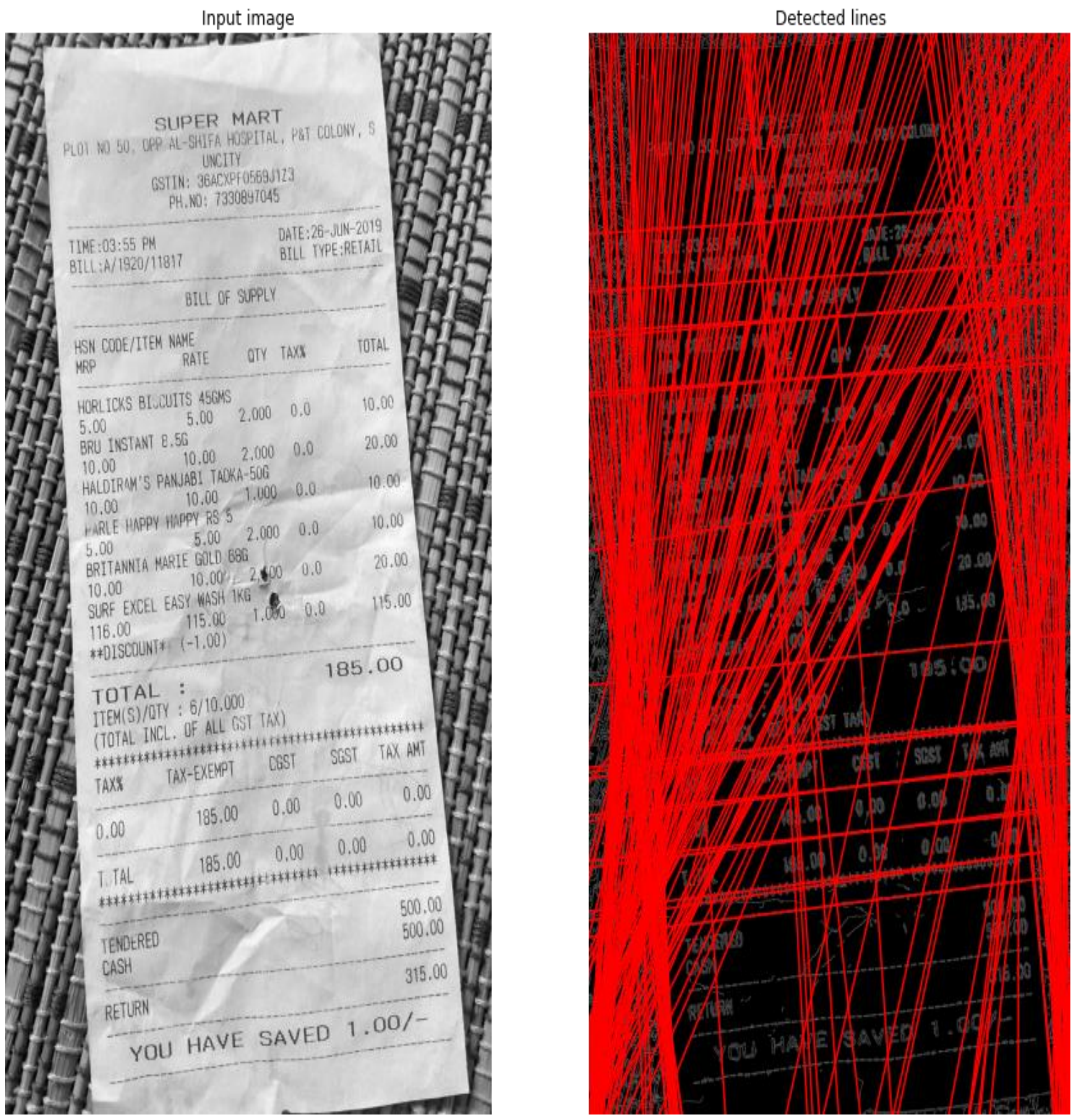
SUPER MART
PLOT NO 50, OPP AL-SHIFA HOSPITAL, P&T COLONY, S
UNCITY
GSTIN: 36ACXPF0569J1Z3
PH.NO: 7330897045

TIME:03:55 PM DATE:26-JUN-2019
BILL:A/1920/11817 BILL TYPE:RETAIL

BILL OF SUPPLY

```
HSN CODE/ITEM NAME
MRP             RATE      QTY   TAX%         TOTAL

HORLICKS BI..CUITS 45GMS
5.00            5.00     2.000  0.0          10.00
BRU INSTANT 8.5G
10.00          10.00     2.000  0.0          20.00
HALDIRAM'S PANJABI TADKA-50G
10.00          10.00     1.000  0.0          10.00
PARLE HAPPY HAPPY RS 5
5.00            5.00     2.000  0.0          10.00
BRITANNIA MARIE GOLD 68G
10.00          10.00     2.000  0.0          20.00
SURF EXCEL EASY WASH 1KG
116.00        115.00     1.000  0.0         115.00
**DISCOUNT**  (-1.00)
                                            185.00
TOTAL  :
ITEM(S)/QTY : 6/10.000
(TOTAL INCL. OF ALL GST TAX)
*************************************************
TAX%     TAX-EXEMPT    CGST     SGST     TAX AMT

0.00       185.00      0.00     0.00       0.00

TOTAL      185.00      0.00     0.00       0.00
*************************************************
                                          500.00
TENDERED                                  500.00
CASH
                                          315.00
RETURN

YOU HAVE SAVED 1.00/-
```

*Figure 5.11: A sample input depicting an image of a receipt take from an angle, with a background, containing wrinkled and slightly faint text*



SUPER MART
PLOT NO 50, OPP AL-SHIFA HOSPITAL, P&T COLONY, S
UNCITY
GSTIN: 36ACXPF0569J1Z3
PH.NO: 7330897045

TIME:03:55 PM DATE:26-JUN-2019
BILL:A/1920/11817 BILL TYPE:RETAIL

BILL OF SUPPLY

| HSN CODE/ITEM NAME | | | | |
|---|---|---|---|---|
| MRP | RATE | QTY | TAX% | TOTAL |
| HORLICKS BI..CUITS 45GMS | | | | |
| 5.00 | 5.00 | 2.000 | 0.0 | 10.00 |
| BRU INSTANT 8.5G | | | | |
| 10.00 | 10.00 | 2.000 | 0.0 | 20.00 |
| HALDIRAM'S PANJABI TADKA-50G | | | | |
| 10.00 | 10.00 | 1.000 | 0.0 | 10.00 |
| PARLE HAPPY HAPPY RS 5 | | | | |
| 5.00 | 5.00 | 2.000 | 0.0 | 10.00 |
| BRITANNIA MARIE GOLD 68G | | | | |
| 10.00 | 10.00 | 2.000 | 0.0 | 20.00 |
| SURF EXCEL EASY WASH 1KG | | | | |
| 116.00 | 115.00 | 1.000 | 0.0 | 115.00 |
| **DISCOUNT** | (-1.00) | | | |

TOTAL : 185.00
ITEM(S)/QTY : 6/10.000
(TOTAL INCL. OF ALL GST TAX)

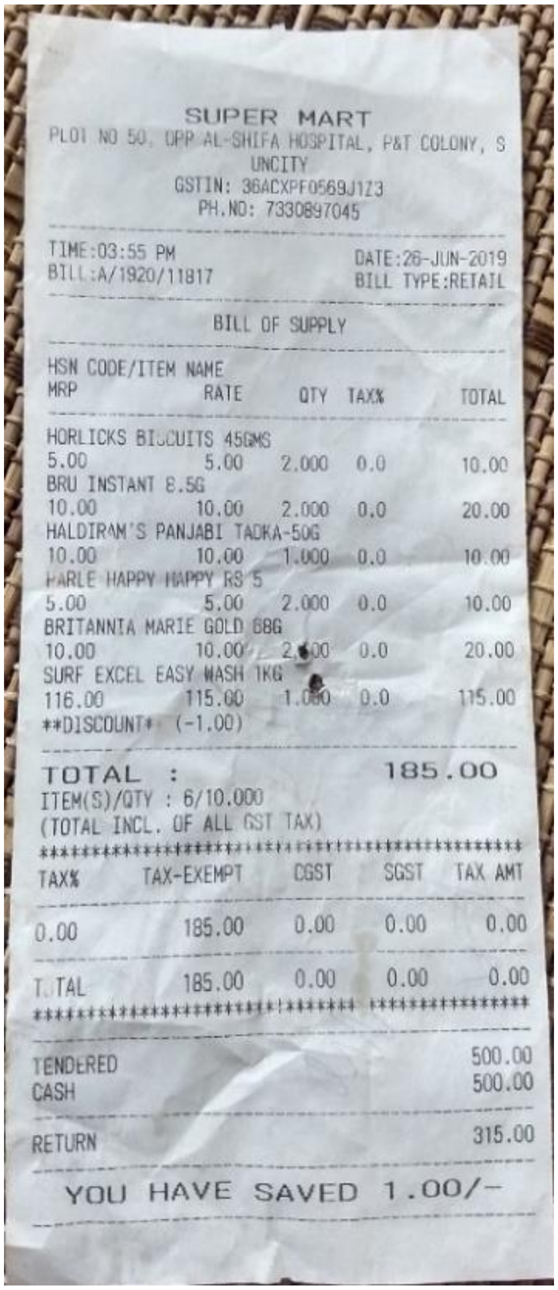
| TAX% | TAX-EXEMPT | CGST | SGST | TAX AMT |
|---|---|---|---|---|
| 0.00 | 185.00 | 0.00 | 0.00 | 0.00 |
| T.TAL | 185.00 | 0.00 | 0.00 | 0.00 |

| | |
|---|---|
| TENDERED | 500.00 |
| CASH | 500.00 |
| RETURN | 315.00 |

YOU HAVE SAVED 1.00/-

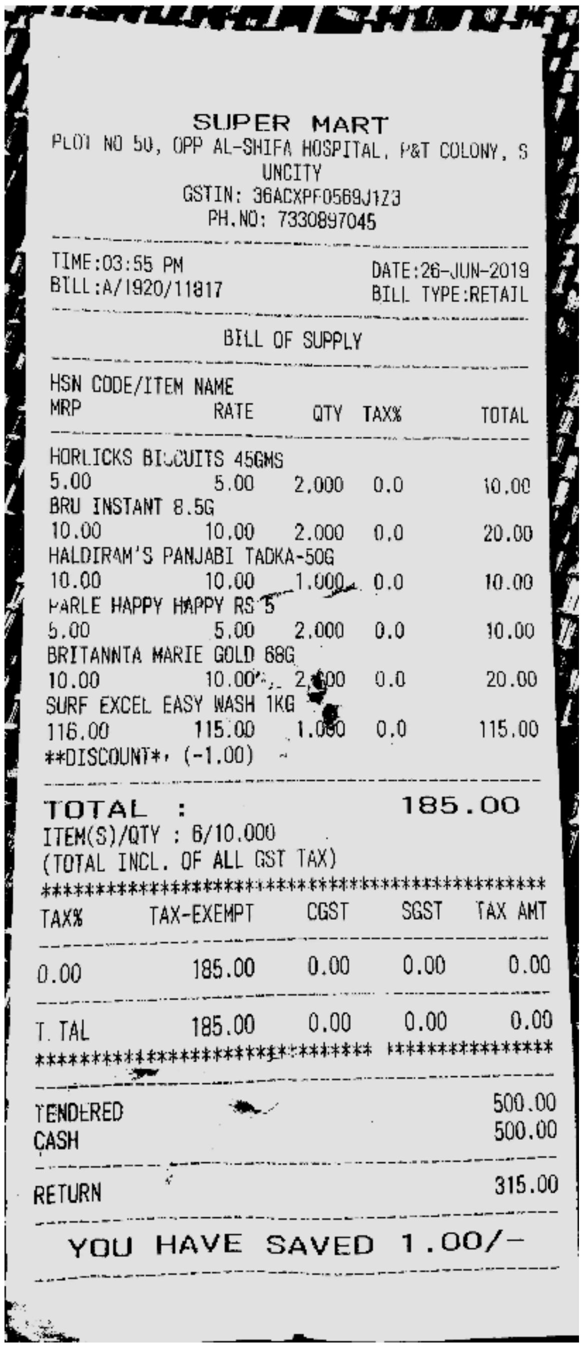
*Figure 6.12: Hough lines detecting that the image is at an angle*

Hough transform is a feature extraction technique that converts an image from Cartesian to polar coordinates which is how it got "transform" in its name. It can be used to detect lines or a set of collinear points on the image. As figure 19 shows, the model is able to detect and place lines passing through the text at the correct angle, as well as finding the orientation it needs to be shifted by. Hough line method also gives us the angle made by the line with the origin.



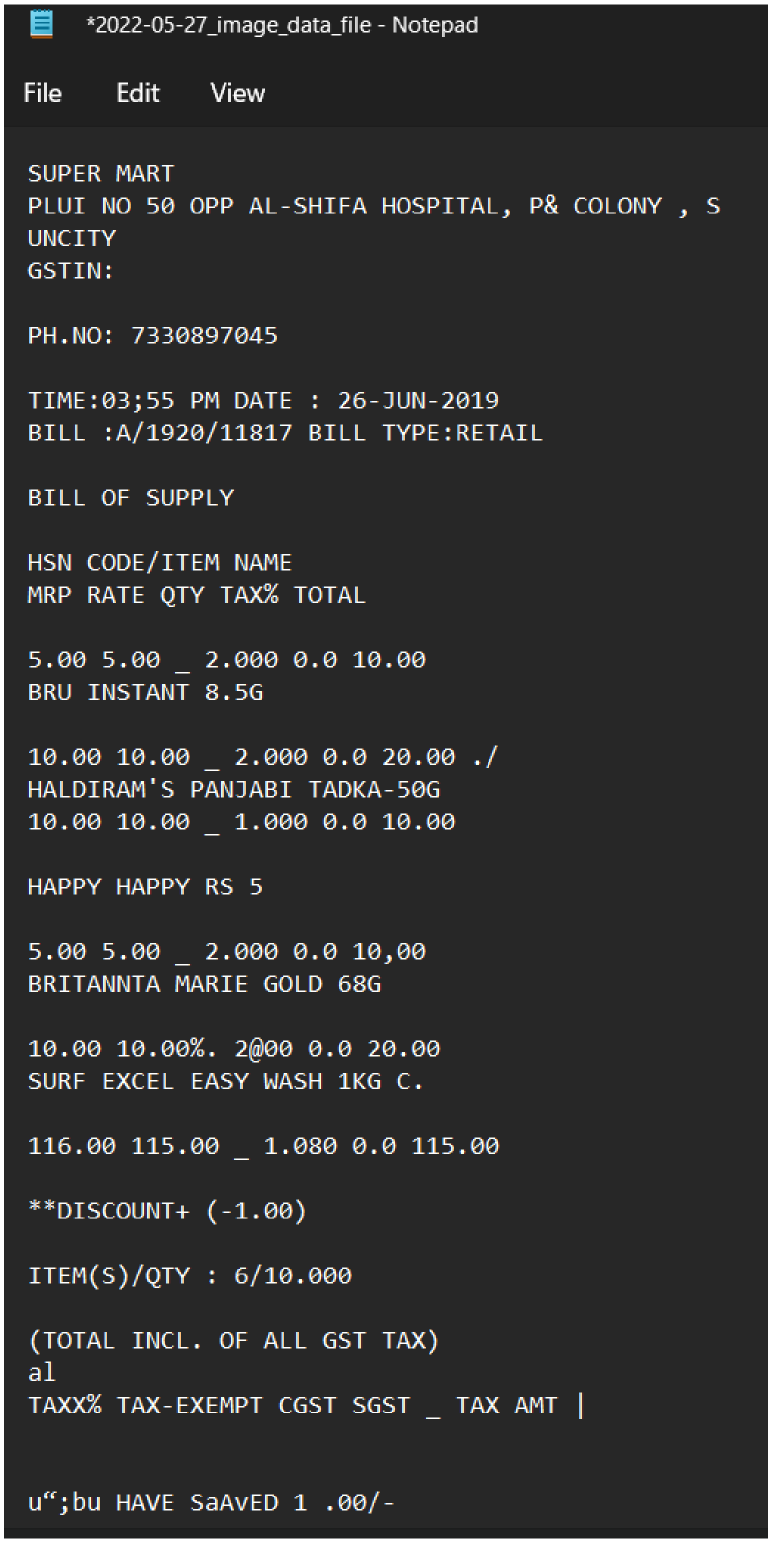
```
                SUPER MART
PLOT NO 50, OPP AL-SHIFA HOSPITAL, P&T COLONY, S
                  UNCITY
          GSTIN: 36ACXPF0569J1Z3
            PH.NO: 7330897045
------------------------------------------------
TIME:03:55 PM                   DATE:26-JUN-2019
BILL:A/1920/11817               BILL TYPE:RETAIL
------------------------------------------------
               BILL OF SUPPLY
------------------------------------------------
HSN CODE/ITEM NAME
MRP             RATE     QTY   TAX%        TOTAL
------------------------------------------------
HORLICKS BISCUITS 45GMS
5.00            5.00   2.000   0.0         10.00
BRU INSTANT 8.5G
10.00          10.00   2.000   0.0         20.00
HALDIRAM'S PANJABI TADKA-50G
10.00          10.00   1.000   0.0         10.00
PARLE HAPPY HAPPY RS 5
5.00            5.00   2.000   0.0         10.00
BRITANNIA MARIE GOLD 68G
10.00          10.00   2.000   0.0         20.00
SURF EXCEL EASY WASH 1KG
116.00        115.00   1.000   0.0        115.00
**DISCOUNT**  (-1.00)
------------------------------------------------
TOTAL  :                         185.00
ITEM(S)/QTY : 6/10.000
(TOTAL INCL. OF ALL GST TAX)
************************************************
TAX%     TAX-EXEMPT     CGST     SGST    TAX AMT
------------------------------------------------
0.00         185.00     0.00     0.00       0.00
------------------------------------------------
TOTAL        185.00     0.00     0.00       0.00
************************************************
------------------------------------------------
TENDERED                                  500.00
CASH                                      500.00
------------------------------------------------
RETURN                                    315.00
------------------------------------------------
        YOU HAVE SAVED 1.00/-
------------------------------------------------
```

*Figure 5.13: The input image after being properly oriented, and the document being detected accurately, having the background being cropped out.*



```
SUPER MART
PLOT NO 50, OPP AL-SHIFA HOSPITAL, P&T COLONY, S
UNCITY
GSTIN: 36ACXPF0569J1Z3
PH.NO: 7330897045
-------------------------------------------------
TIME:03:55 PM                 DATE:26-JUN-2019
BILL:A/1920/11817             BILL TYPE:RETAIL
-------------------------------------------------
BILL OF SUPPLY
-------------------------------------------------
HSN CODE/ITEM NAME
MRP           RATE      QTY   TAX%        TOTAL
-------------------------------------------------
HORLICKS BI.CUITS 45GMS
5.00          5.00     2.000  0.0         10.00
BRU INSTANT 8.5G
10.00         10.00    2.000  0.0         20.00
HALDIRAM'S PANJABI TADKA-50G
10.00         10.00    1.000  0.0         10.00
PARLE HAPPY HAPPY RS 5
5.00          5.00     2.000  0.0         10.00
BRITANNTA MARIE GOLD 68G
10.00         10.00    2.000  0.0         20.00
SURF EXCEL EASY WASH 1KG
116.00        115.00   1.000  0.0        115.00
**DISCOUNT** (-1.00)
-------------------------------------------------
TOTAL :                              185.00
ITEM(S)/QTY : 6/10.000
(TOTAL INCL. OF ALL GST TAX)
*************************************************
TAX%     TAX-EXEMPT    CGST     SGST    TAX AMT
-------------------------------------------------
0.00       185.00      0.00     0.00     0.00
-------------------------------------------------
T TAL      185.00      0.00     0.00     0.00
*************************************************
-------------------------------------------------
TENDERED                                 500.00
CASH                                     500.00
-------------------------------------------------
RETURN                                   315.00
-------------------------------------------------
YOU HAVE SAVED 1.00/-
-------------------------------------------------
```

*Figure 5.14: The text on the image is isolated and enhanced to increase the accuracy of text detection and extraction*



```
SUPER MART
PLUI NO 50 OPP AL-SHIFA HOSPITAL, P& COLONY , S
UNCITY
GSTIN:

PH.NO: 7330897045

TIME:03;55 PM DATE : 26-JUN-2019
BILL :A/1920/11817 BILL TYPE:RETAIL

BILL OF SUPPLY

HSN CODE/ITEM NAME
MRP RATE QTY TAX% TOTAL

5.00 5.00 _ 2.000 0.0 10.00
BRU INSTANT 8.5G

10.00 10.00 _ 2.000 0.0 20.00 ./
HALDIRAM'S PANJABI TADKA-50G
10.00 10.00 _ 1.000 0.0 10.00

HAPPY HAPPY RS 5

5.00 5.00 _ 2.000 0.0 10,00
BRITANNTA MARIE GOLD 68G

10.00 10.00%. 2@00 0.0 20.00
SURF EXCEL EASY WASH 1KG C.

116.00 115.00 _ 1.080 0.0 115.00

**DISCOUNT+ (-1.00)

ITEM(S)/QTY : 6/10.000

(TOTAL INCL. OF ALL GST TAX)
al
TAXX% TAX-EXEMPT CGST SGST _ TAX AMT |


u“;bu HAVE SaAvED 1 .00/-
```

*Figure 5.15: The output data extracted for the input image*

## 5.3. Result Analysis

We performed a comparative study of previous works and the limitations observed in each of these works. We then summarize our method and that the results that we have observed not only reached the objectives we have set but have also addressed a majority of the limitations seen in previous works in the field.

In Event Info Extraction from Flyers [11], the images that were considered in the dataset were all taken from a perpendicular upright view, the illumination of the images not being uniform has caused a higher error rate.

This paper, Text extraction using OCR: A Systematic Review [1], only focused on skew correction, all the images were of high quality and hence they didn't require to consider the use of pre-processing methods.

The methodology used in the paper, Improve OCR Accuracy with Advanced Image Preprocessing using Machine Learning with Python [2], does not cover images with uneven brightness, watermarks, or different fonts.

A reinforcement learning approach is used in "Improving the Accuracy of Tesseract 4.0 OCR Engine Using Convolution-Based Pre-processing" [16] and does not guarantee that local optimums will be avoided each time. The algorithm gets stuck on kernel configuration.

In the paper, Text Recognition from Images: A Study [3], their methodology is targeted at, and successfully performs noise removal by applying Gaussian filter and mean filter.

BLSTM-Bidirectional Long Short-Term Memory is used by the authors of the paper titled, Recognition of Printed Devanagari Text Using BLSTM Neural Network [4], which did not require segmentation. The also observed a very low accuracy.

For Information extraction, in the paper titled, System that automatically converts word into text [10], they concluded that an n-gram dictionary can be used so that words may be correctly identified that are not in the dictionary of geographical names.

A Study on Various Image Processing Techniques [6], in this paper the authors performed removal of noise using Laplacian and Harr Filtering only.

The algorithm used in "Robust Text Extraction in Images for Personal Event Planner" [12], is not tested for event information taken from handwritten images and complex fonts of printed text present in the images.

The authors of paper titled, Text recognition on images from social media [5], worked on a dataset collected from social media where the images were of high quality and of only printed text.

The Deep Learning Model for Text Recognition in Images [7], suggests the use of Convolutional Neural Network and Long Short-term Memory for text recognition.

In the paper, Design and implementation of android application to extract text from images by using tesseract for English and Hindi [13], the user edits or crops the image based on what text he wants; the process isn't automated.

The pre-processing techniques that were included in this paper title, Analytical Review of Preprocessing Techniques for Offline Handwritten Character Recognition [15], are contrast stretching, noise removal techniques, normalization and segmentation. Though it was a detailed pipeline, the accuracy of the pre-processed image doesn't improve the text extraction accuracy.

The authors of the paper titled, Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models [14], proposed a new method in the form of pre-processing and recognition which are both based on ANNs.

In the paper, Steps Involved in Text Recognition and Recent Research in OCR; A Study [8], outlined only a text recognition flow and suggested classifiers such as ANN and SVM and didn't focus on pre-processing the images or how the classifiers can be implemented.

Our proposed solution has an image pre-processing pipeline customises the methods it applies based on problems detected in the image such as gradient brightness, skewness, blurriness, noisy images etc. We pre-process the image by sharpening, brightening, deblurring, noise removal, de-skewing, and detecting documents from the background. We then detect and extract printed text from images efficiently and additionally convert the data into an audio file.

Images used in our dataset contained a count of words ranging from **800 to 1500 per document**. We observed an **average error rate** of **11% at word level**.

The entire processing of the image to the conversion to speech takes about **3-5 seconds**.

# 6. CONCLUSION

In this report, the objective of the project is explained, the task is introduced - including an explanation of the approach and the solution is illustrated. To achieve better performance, carefully chosen and placed elaborate image processing techniques are used, to increase the probability of retrieving desired text from the OCR engine. An NLP model converts raw data that may include misspelled words to structured data that can be downloaded as a copy. Acting as an additional support to the visually impaired, the system also converts the structured data to speech data in the form of an audio file that can be downloaded.

## Limitations

At this stage, we are aware of the aspects of the project that are yet to be fulfilled. The points listed below show areas that require improvement, which will be rendered in the near future.

- Handwritten characters are not accurately extracted. Although the preprocessing pipeline retains, as well as enhances and sharpens the characters, the OCR engine used does not recognize the characters.
- We observe an error rate of 11% on an average, an if this system were to be used in institutions, any form of error isn't desirable.

## Future Scope

More features that can be added to this application are listed below. These will be developed in the near future, and the project will be kept alive with regular updates and modifications.

- Training and implementing a model to recognize and extract handwritten characters. In a document containing both printed and hand written content, we want to be able to list the data in handwritten format in the same order it appears alongside and amidst the printed text.
- The entire process starting from uploading the image to extracting downloadable data takes 3 seconds on an average, and up to 5 seconds for images with more content. We want to decrease this wait time for the user down to 1 second maximum.

# REFERENCES

[1]. Mittal, Rishabh; Garg, Anchal, “Text extraction using OCR: A Systematic Review”, 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020, pp 357–362.

[2]. Sanjeev Kumar, Mahika Sharma, Kritika Handa, Rishika Jaiswal, “Dan”, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-9 Issue-7, May 2020.

[3]. Sahana K Adyanthaya, “Text Recognition from Images: A Study”, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Volume-8 Issue-13, 2020.

[4]. Naveen Sankaran and C.V Jawahar, “Recognition of Printed Devanagari Text Using BLSTM Neural Network”, 21st International Conference on Pattern Recognition (ICPR), November 11-15, 2012. Tsukuba, Japan.

[5]. M.S. Akopyan, O.V. Belyaeva, T.P. Plechov and D.Y. Turdakov, “Text recognition on images from social media”, Ivannikov Memorial Workshop (IVMEM), 2019.

[6]. Dr. PL Chithra, P Bhavani, P., “A Study on Various Image Processing Techniques”, International Journal of Emerging Technology and Innovative Engineering (ISSN (print): 2394 – 6598) Volume 5, Issue 5, May 2019.

[7]. Anupriya Shrivastava, Amudha J., Deepa Gupta, Kshitij Sharma, “Deep Learning Model for Text Recognition in Images”, 10th ICCCNT 2019 July 6-8, 2019, IIT - Kanpur, Kanpur, India.

[8]. K. Karthick, K.B. Ravindrakumar, R. Francis, S. Ilankannan, “Steps Involved in Text Recognition and Recent Research in OCR; A Study”, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8, Issue-1, May 2019.

[9]. Sai Abhishikth Ayyadevara, P N V Sai Ram Teja, Bharath K P Rajesh Kumar M, “Handwritten Character Recognition Using Unique Feature Extraction Technique”, International Research Journal of Modernization in Engineering Technology and Science, Volume-3 Issue-10, Jan 2020.

[10]. Karen Kukich, “System that automatically converts word into text”, ACM Computing Surveys, Vol. 24, No. 4, December 1992.

[11]. Yang Zhang, Hao Zhang, HaoranLi, “Event Info Extraction from Flyers”, 2021.

[12]. Bhaskar, L., & Ranjith, R., “Robust Text Extraction in Images for Personal Event Planner”, 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) July 1-3, 2020, IIT-Kharagpur, Kharagpur, India.

[13]. Brijeshkumar Y. Panchal and Gaurang Chauhan, “Design and implementation of android application to extract text from images by using tesseract for English and Hindi”, 3rd International Scientific Conference of Engineering Sciences and Advances Technologies (IICESAT), Journal of Physics: Conference Series, Volume 1973, 4-5 June 2021.

[14]. Salvador España-Boquera, Maria J. C. B., Jorge G. M. and Francisco Z. M., “Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, No. 4, April 2011.

[15]. K. Gaurav and Bhatia P. K., “Analytical Review of Preprocessing Techniques for Offline Handwritten Character Recognition”, 2nd International Conference on Emerging Trends in Engineering & Management, ICETEM, 2013.

[16] Sporici, Dan & Cușnir, Elena & Boiangiu, Costin-Anton. (2020). Improving the Accuracy of Tesseract 4.0 OCR Engine Using Convolution-Based Preprocessing. Symmetry. 12. 715. 10.3390/sym12050715.

# APPENDIX I

## CODE

```
import cv2

from .image_helper_functions import *

import numpy as np

from skimage.transform import hough_line, hough_line_peaks

from skimage.transform import rotate

from skimage.feature import canny

from skimage.io import imread

from skimage.color import rgb2gray

from scipy.stats import mode


def grayscale(image, path):

    if(len(image.shape)>=3):

        # convert to grayscale

        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

        save_image(image,"_grayscaled.png",path)

    return image


def edge_detection(image):

    kernel = np.ones((5,5),'uint8')

    image = cv2.dilate(image,kernel,iterations=1)

    edges = cv2.Canny(image, 100, 70)

    return edges
```

```
def morphology(image):
    # threshold
    thresh = cv2.threshold(image, 190, 255, cv2.THRESH_BINARY)[1]
    # apply morphology
    kernel = np.ones((7,7), np.uint8)
    morph = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)
    kernel = np.ones((9,9), np.uint8)
    morph = cv2.morphologyEx(morph, cv2.MORPH_ERODE, kernel)
    return morph


def contour_detection(image):
    # convert to grayscale
    gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    # threshold
    thresh = cv2.threshold(gray, 190, 255, cv2.THRESH_BINARY)[1]\
    # apply morphology
    kernel = np.ones((7,7), np.uint8)
    morph = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)
    kernel = np.ones((9,9), np.uint8)
    morph = cv2.morphologyEx(morph, cv2.MORPH_ERODE, kernel)
    # get largest contour
    contours = cv2.findContours(morph, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
    contours = contours[0] if len(contours) == 2 else contours[1]
    area_thresh = 0
    for c in contours:
        area = cv2.contourArea(c)
        if area > area_thresh:
            area_thresh = area
            big_contour = c

    # get bounding box
    x,y,w,h = cv2.boundingRect(big_contour)
    # draw filled contour on black background
    mask = np.zeros_like(gray)
    mask = cv2.merge([mask,mask,mask])
    cv2.drawContours(mask, [big_contour], -1, (255,255,255), cv2.FILLED)
    # apply mask to input
    result1 = image.copy()
    result1 = cv2.bitwise_and(result1, mask)
    # crop result
    result2 = image[y:y+h, x:x+w]
    return result2

def skew_correction(image):
    # convert to edges
    # Classic straight-line Hough transform between 0.1 - 180 degrees.
    tested_angles = np.deg2rad(np.arange(0.1, 180.0))
    h, theta, d = hough_line(edge_detection(image), theta=tested_angles)
    # find line peaks and angles
    accum, angles, dists = hough_line_peaks(h, theta, d)
    # round the angles to 2 decimal places and find the most common angle.
    most_common_angle = mode(np.around(angles, decimals=2))[0]
    # convert the angle to degree for rotation.
    skew_angle = np.rad2deg(most_common_angle - np.pi/2)
    print(skew_angle)
    # skewed_img=rotate(image, skew_correction(image), cval=1)
    return skew_angle


def blur_correction(image):
    sharpen_kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
    sharpen_img = cv2.filter2D(image, -1, sharpen_kernel)
    return sharpen_img


def shadow(image):
    dilated_img = cv2.dilate(image, np.ones((7,7), np.uint8))
    bg_img = cv2.medianBlur(dilated_img, 21)
    diff_img = 255 - cv2.absdiff(image, bg_img)
    norm_img = diff_img.copy() # Needed for 3.x compatibility
    cv2.normalize(diff_img, norm_img, alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8UC1)
    _, thr_img = cv2.threshold(norm_img, 230, 0, cv2.THRESH_TRUNC)
    cv2.normalize(thr_img, thr_img, alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8UC1)
    return thr_img
```

```
def denoise(image):
    dst = cv2.fastNlMeansDenoisingColored(image, None, 11, 6, 7, 21)
    denoised_img = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)
    return denoised_img
```

# APPENDIX II

## GUI CODE

```
import sys

import os

import tkinter as tk

from tkinter import scrolledtext, ttk, messagebox

from tkinter.filedialog import askopenfilename

import subprocess

from multiprocessing import Process, Queue

from queue import Empty

from turtle import width

import datetime


today = datetime.date.today()

save_path = os.path.join(os.getcwd(),"data\\"+str(today)+'\\')

global file_names

file_names = ''


if not os.path.exists(save_path):

    os.makedirs(save_path)

if not os.path.exists(save_path+'images\\'):

    os.makedirs(save_path+'images\\')


DELAY1 = 60

DELAY2 = 20
```

```
q = Queue()


class OCRNets(tk.Tk):
    def __init__(self):
        # initial window after running the code
        super().__init__()
        self.filepath = False
        self.title("OCRNets")
        self.tk.call('wm', 'iconphoto', self._w, tk.PhotoImage(file='logo.png'))
        self.geometry("1200x550+100+100")


        self.label_1 = tk.Label(self, text="Welcome to OCRNets!",font=(16))
        self.label_1.place(x=600, y=100,anchor='center')


        self.label_2 = tk.Label(self, text="Please upload you image in the next page for
text extraction", font=("bold", 11))
        self.label_2.place(x=600, y=140,anchor='center')


        self.btn_1 = ttk.Button(self, text="Proceed",
width='20',command=lambda:destroy_widgets() or self.upload_page())
        self.btn_1.place(x=600,y=180,anchor='center')


        def destroy_widgets():
            self.label_1.destroy()
            self.label_2.destroy()
            self.btn_1.destroy()
    def upload_page(self):
        self.label_1 = tk.Label(self,text="Please upload an image of the document you would like to extract text from",font=('bold',12))
        self.label_1.place(x=600,y=100,anchor="center")

        self.btn_1 = ttk.Button(self, text='Open Image',width=25, command=lambda: self.open_file() or destroy_widgets() or self.upload_verification_page())
        self.btn_1.place(x=600,y=140,anchor="center")

        def destroy_widgets():
            self.label_1.destroy()
            self.btn_1.destroy()

    def open_file(self):
        self.filepath = askopenfilename()

    def upload_verification_page(self):
        if self.filepath:
            self.label_1 = tk.Label(self,text="The File you have uploaded is: "+str(self.filepath),font=('bold',11))
            self.label_1.place(x=600,y=100,anchor="center")

            self.btn_1 = ttk.Button(self, text='Change Uploaded File', command=lambda: destroy_widgets() or self.upload_page())
            self.btn_1.place(x=500,y=150,anchor='center')
            self.btn_2 = ttk.Button(self,text="Use this Image",command=lambda: destroy_widgets() or self.extract_text() or self.onStart())
            self.btn_2.place(x=700,y=150,anchor="center")
        else:
            self.upload_page()


        def destroy_widgets():
            self.label_1.destroy()
            self.btn_1.destroy()
            self.btn_2.destroy()


    def extract_text(self):
        self.label_1 = tk.Label(self, text='Extracting text from your image......')
        self.label_1.place(x=500, y=100,anchor='center')


        self.progress_bar = ttk.Progressbar(self, mode='indeterminate', length=150)
        self.progress_bar.place(x=700,y=100,anchor='center')


        self.txt = scrolledtext.ScrolledText(self,width = 80, height = 7)
        self.txt.place(x=600,y=190,anchor='center')
        self.txt.insert('end', "Logs...."+'\n')


        self.btn_1 = ttk.Button(self, text="Next ", command=lambda: destroy_widgets() or self.display_and_download())
        self.btn_1.place(x=950,y=280,anchor='center')
        def destroy_widgets():
            self.label_1.destroy()
            self.progress_bar.destroy()
            self.txt.destroy()
            self.btn_1.destroy()

    def display_and_download(self):
        self.label_1 = tk.Label(self,text="You can find the audio and text files saved
at:"+str(save_path),font=('bold',11))
        self.label_1.place(x=600,y=100,anchor="center")

    def onStart(self):
        # self.startBtn.config(state=tk.DISABLED)
        self.txt.delete("1.0", tk.END)
        self.process_ = Process(target=image_digitization, args=(q, self.filepath))
        self.process_.start()
        self.progress_bar.start(DELAY2)
        self.after(DELAY1, self.onGetValue)

    def onGetValue(self):
       if (self.process_.is_alive()):
        self.after(DELAY1, self.onGetValue)
        return
       else:
           try:
                self.txt.insert('end', q.get(0))
                self.txt.insert('end', "\n")
                self.progress_bar.stop()
                # self.text_file_name = file_names[0].split('\\')
                # self.audio_file_name = file_names[1].split('\\')
                # print(self.text_file_name, self.audio_file_name)
            except Empty:
                print("queue is empty")


def image_digitization(q, filepath):
    global file_names
    text = subprocess.getstatusoutput('Python image_to_text.py '+filepath+' '+save_path)
    print(type(text[1]))
    file_names = text[1].split('\n')
    print(file_names)
    q.put(text[1])


if __name__ == "__main__":
    OCRNets().mainloop()
```

# APPENDIX III

## USER REQUIREMENTS

The user needs to upload an image. This image is processed by a model at backend. The user is also required to download and install the required packages [refer Appendix-IV].

## SOFTWARE REQUIREMENTS

- 32-bit or 64-bit operating system
- Python (3.6 or above)
- Numpy, OpenCV, PIL (pillow), Tesseract-OCR, pytesseract, sklearn, nltk, Django, Tkinter

## HARDWARE REQUIREMENTS

- Processor: i3 10th Gen or above
- Ram: Minimum 4GB RAM. 8GB RAM is suggested for speed and smooth functioning
- Disk Space: 10 GB minimum

# APPENDIX IV

## Set up guide for the application

To install the requirements and run the code the following steps must be followed:

1. Install Python 3.6 or above (https://www.python.org/downloads/)

2. Open cmd or terminal and run the following:

```
pip install tesseract-ocr

pip install scikit-learn

pip install opencv-python

pip install pillow

pip install pytesseract

pip install nltk

pip install Django
```

3. Once all the requirements are downloaded open the project folder on cmd or terminal and run - Python manage.py runserver This command will run the application on localhost:8000

4. On the browser go to http://localhost:8000/

The web page is opened, to go to your dashboard, click on Get Started.