CONVERTING TYPED OR HANDWRITTEN DOCUMENTS, SCANNED OR PHOTO IMAGES TO LEGIBLE TEXT

A MINI PROJECT REPORT

submitted

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

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

G.HARI PRIYA
(17B81A0565)
CH.INDUPRIYA
(17B81A0573)
K.KALYAN
(17B81A0585)

Under the guidance of

Dr. R.K. SELVAKUMAR

Professor/CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NBA, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

June 2020

ACKNOWLEDGEMENTS

The satisfaction that accomplishes the successful completion of any tasks would be incomplete without the mention of the people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project.

It is great pleasure to convey our profound sense of gratitude to our principal **Dr. Nayanathara K.S**, Vice Principal **Prof. L.C. Siva Reddy, Dr. K.Venkateswara Rao**, Head of the CSE Department, CVR college of Engineering for having been kind enough for arranging the necessary facilities for executing the project in the college.

We would like to express my sincere gratitude to my guide, **Dr. R. K. Selvakumar**, Professor, CSE Dept., CVR College of Engineering, whose guidance and valuable suggestions have been indispensable to bring about the successful completion of our project.

We wish a deep sense of gratitude and heartfelt thanks to management for providing excellent lab facilities and tools. Finally, we thank all those whose guidance helped us in this regard.

Yours Sincerely

G.Hari Priya-17B81A0565

CH.Indupriya-17B81A0573

K.Kalyan-17B81A0585

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Cherabuddi Education Society's
CVR COLLEGE OF ENGINEERING
(An Autonomous Institution)
ACCREDITED BY NBA, NAAC 'A' Grade
(Approved by AICTE & Government of Telangana and Affiliated to JNTU Hyderabad)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M), R.R.District.
Web: http://www.cvr.ac.in, email: info@cvr.ac.in
Ph : 08414 – 252222, 252369, Office Telefax : 252396,
Principal : 252396 (O)

CERTIFICATE

This is to certify that the project entitled **"CONVERTING TYPED OR HANDWRITTEN DOCUMENTS, SCANNED OR PHOTO IMAGES TO LEGIBLE TEXT "** that is being submitted by **G.HARI PRIYA** (17B81A0565), **CH.INDUPRIYA** (17B81A0573), **K.KALYAN** (17B81A0585) in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** to the **CVR College of Engineering**, is a record of bonafide work carried out by them under my guidance and supervision during the year 2019-2020.

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

Signature of the project guide,

Dr. R.K.SelvaKumar,
Professor,
CSE Department

Signature of the HOD

Dr. K. Venkateswara Rao,
Head Of Department(CSE),
CVR College Of Engineering

External Examiner

ABSTRACT

Optical character recognition, usually abbreviated to OCR, is the electronic conversion of scanned or photographed images of typewritten or printed text into machine-encoded/computer-readable text. It is widely used as a form of data entry from some sort of original paper data source, whether passport documents, invoices, bank statement, receipts, business card, mail, or any number of printed records. 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 machine translation, text-to-speech, key data extraction and text mining.

The aim of this project is to develop an Optical Capture Recognition Interface (OCR). Scanned text documents, pictures stored in desktops having windows as operating system, are the main focus of this application. Using this application confidential information is extracted, digitalized and stored in database in a structured manner. Hence, this proposed system mainly focuses on retrieval of information from PAN cards, and then digitalize the significant information.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In the present world, there's an ever-increasing demand for the software systems to acknowledge characters in computing system when information is scanned through paper documents. There's an enormous demand in storing the knowledge available in these paper documents into a memory disk and then reusing the information by the process of retrieval. One simple way to store the information in paper documents into a computer system is to first scan the document and then store them as images. To reuse the information it requires manual work such as reading line-by-line or word-by-word which is a tedious task. The concept of storing the contents of paper documents in computer storage and then reading, searching the content is called **Document Processing**. For this process we need a software system called Character Recognition System. This process is also called **Document Image Analysis.** To perform document image analysis, which transforms the images to digital format there is a need to develop character recognition software system. Optical Character Recognition (OCR) is one of the widely used techniques to recognize characters.

OCR systems transform initially scanned document which is a two-dimensional image of text, that could contain machine printed or handwritten text from its image representation into machine-readable text. OCR as a process generally consists of several sub-processes to perform as accurately as possible. Accuracy of OCR can be dependent on text preprocessing and segmentation algorithms. Sometimes it is difficult to retrieve text from the image because of different size, style, orientation, complex background of image etc. In OCR software its main aim is to identify and capture all the unique words using different languages from written text characters.

1.1 Motivation

There are many techniques which are being used to recognize different language characters present in the image. One such widely used technique is Optical Character Recognition (OCR). OCR is like combination of eye and mind of human body. An eye can view the text from the images but actually the brain processes as well as interprets that extracted text read by eye. In development of computerized OCR system, few problems can occur. First: there is very little visible difference between some letters and digits for

computers to understand. For example it might be difficult for the computer to differentiate between digit “0” and letter “o”. Second: It might be very difficult to extract text, which is embedded in very dark background or printed on other words or graphics.

The problem of procuring maximum accuracy by humans is deficit and burdensome. So if this task is computerized it not only curtails error rate but also saves the time and aid to efficient retrieval in future.

1.2 Problem Statement

To create a web based application where the typed or printed documents or images are converted into legible text. The required data from the image is extracted and re-structured in a tabular form. Here the user can access the critical data whenever required only for read operation. The critical data cannot be modified thereby protecting its integrity. This way we can digitalize the data through image processing and cutting edge technologies.

1.3 Project Report Organization

i. This report is divided into 5 chapters after this introductory chapter.
ii. Chapter 2 gives insights about proposed Digitalization of critical data, introduces characteristics of the problem and design challenges.
iii. Chapter 3 summarizes functional, non-functional requirements and system requirements along with software and hardware specifications.
iv. Chapter 4 deals with analysis and design of the proposed model which includes system architecture and technology description.
v. Chapter 5 encloses Implementation of the proposed model and testing with different scenarios.
vi. Chapter 6 includes conclusion and future work along with references.

2. PROPOSED WORK – CRITICAL DATA EXTRACTION FROM DIGITAL IMAGES (CDEDI)

2.1 Characteristics of the Problem

OCR also called Optical Character Recognition is a system that provides a full alphanumeric recognition of printed or handwritten characters at electronic speed by simply scanning the form. Forms containing characters images can be scanned through scanner and then recognition engine of the OCR system interpret the images and turn images of handwritten or printed characters into ASCII data (machine-readable characters). Therefore, OCR allows users to quickly automate data capture from forms, eliminate keystrokes to reduce data entry costs and still maintain the high level of accuracy required in forms processing applications. The technology provides a complete form processing and documents capture solution.

Usually, OCR uses a modular architecture that is open, scalable and workflow controlled. It includes forms definition, scanning, image pre-processing, and recognition capabilities. This desktop application will allow its users to retrieve text from these documents to refer it to perform other actions. For example, people can scan their PAN Card, and store its data.

Optical Character Recognition, or OCR, is a technology that enables us to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera or phone into editable and searchable data. This technology is very useful since it saves time without the need of manual entry. It can perform the action in few minutes. It is able to recognize text in images and convert it into editable text by going through a simplified process.

2.2 Design Challenges

For good quality and high accuracy character recognition, OCR techniques expect high quality or high resolution images with some basic structural properties such as high differentiating text and background. The way images are generated is an important

and determining factor in the accuracy and success of OCR, since this often affects the quality of images dramatically. Usually OCR with images produced by scanners gives high accuracy and good performance. In contrast, images produced by cameras usually are not as good of an input as scanned images to be used for OCR due to the environmental or camera related factors. Numerous errors might emerge, which are clarified as follow.

1. There is very little visible difference between some letters and digits for computers to understand. For example it might be difficult for the computer to differentiate between digit “0” and letter “o”.
2. It might be very difficult to extract text, which is embedded in very dark background or printed on other words or graphics.
3. Skewing of text lines from their unique orientation may cause great degree of poor results.

For the most part connected with photography, there are two kinds of obscure which is:

i) out of focus obscure
ii) movement obscure

2.3 Proposed Solution

The purpose of this project is to efficiently extract the text contained in a document (converted into an image), buffer it in a JSON and then store it in xsl sheet. There are many engines that serve as solution of which Tesseract OCR is widely used. Considering this, Tesseract OCR Engine is used to accomplish this project. We have tested high, medium and low quality images in the project. The accuracy of the data extracted depends on the resolution of the input image.

The proposed system works in the following manner in three phases:

Phase - I

1. Initially, the document is scanned and stored in a supported image format.
2. If the image contains any rotated block of text at an unknown angle, there is a need to correct the text. This can be done by de-skewing.
3. De-skewing involves finding the angle and rotating the image to correct for the skew.

4. The image obtained after the above step is then passed as input to the next phase.

Phase – II

5. The image then undergoes morphology which is a method of processing digital images on the basis of shape.
6. The morphed image is then pre-processed to achieve better results.

Phase – III

7. Localization: Identifying the area of interest and drawing the bounding boxes around the region.
8. Refinement: The data may contain undesirable characters for some fields, then those characters are modified or deleted accordingly.
9. Extraction and storing: Extracting the data from region of interest, buffering into a JSON file and then organizing in xsl sheet.

3. REQUIREMENTS AND SPECIFICATIONS

3.1 Software Requirements

3.1.1 Functional Requirements

i. The system should process the input given by user only if it is an image file.

ii. System shall show the error message to the user when input given is not in the required format

iii. System should detect characters present in the image.

iv. System should retrieve characters present in the image and store them in database

3.1.2 Non Functional Requirements

i. **Availability:** This system will retrieve the handwritten or typed text regions.

ii. **Functionality:** This software will deliver on the functional requirements mentioned.

iii. **Reliability:** This software will work reliably for low resolution images and not for graphical images.

iv. **Learning ability:** Easy to use.

3.2 System Specifications

3.2.1 Software Specifications

Table 3.2.1 Software Specifications

Operating System	Windows
Language	Python 3.x
Libraries	i. argparse ii. cv2 iii. csv iv. ftfy v. io vi. json vii. numpy viii. os ix. PIL x. pytesseract xi. sys
Software(OCR Engine)	Tesseract

3.2.2Hardware Specifications:

Table 3.2.2 Hardware Specifications

Processor	i3 and above
RAM	32MB(min)
Hard Disk	4GB
Scanner	300 dpi

4. ANALYSIS AND DESIGN

4.1 Use Case Diagram- CDEDI

The Figure 4.1 shows the Critical Data Extraction Processes in Digital Image for customer ID/Check in the system.

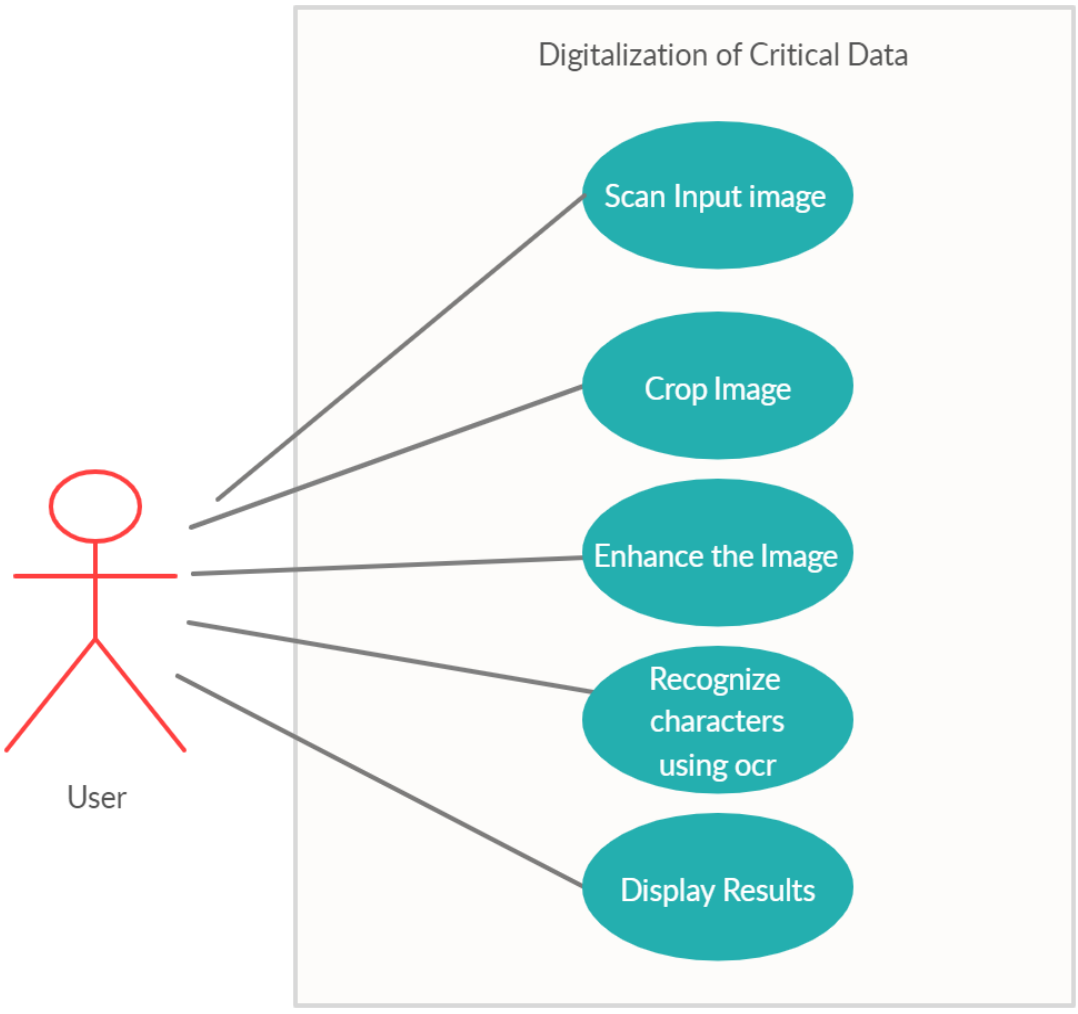

Figure 4.1 Customer ID Card Recognition Use case diagram

4.2 Class Diagram

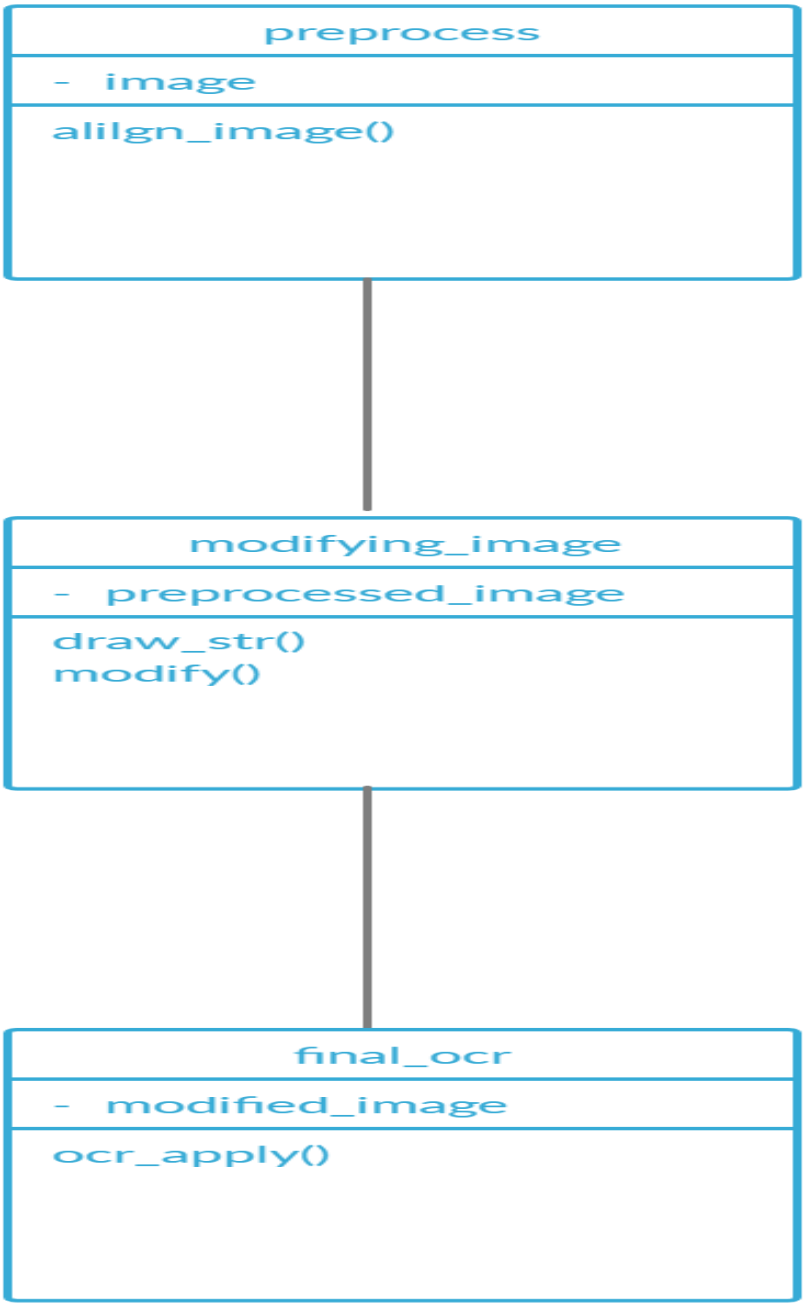

Figure 4.2 Class Diagram for **CDEDI**

4.3 Activity Diagram

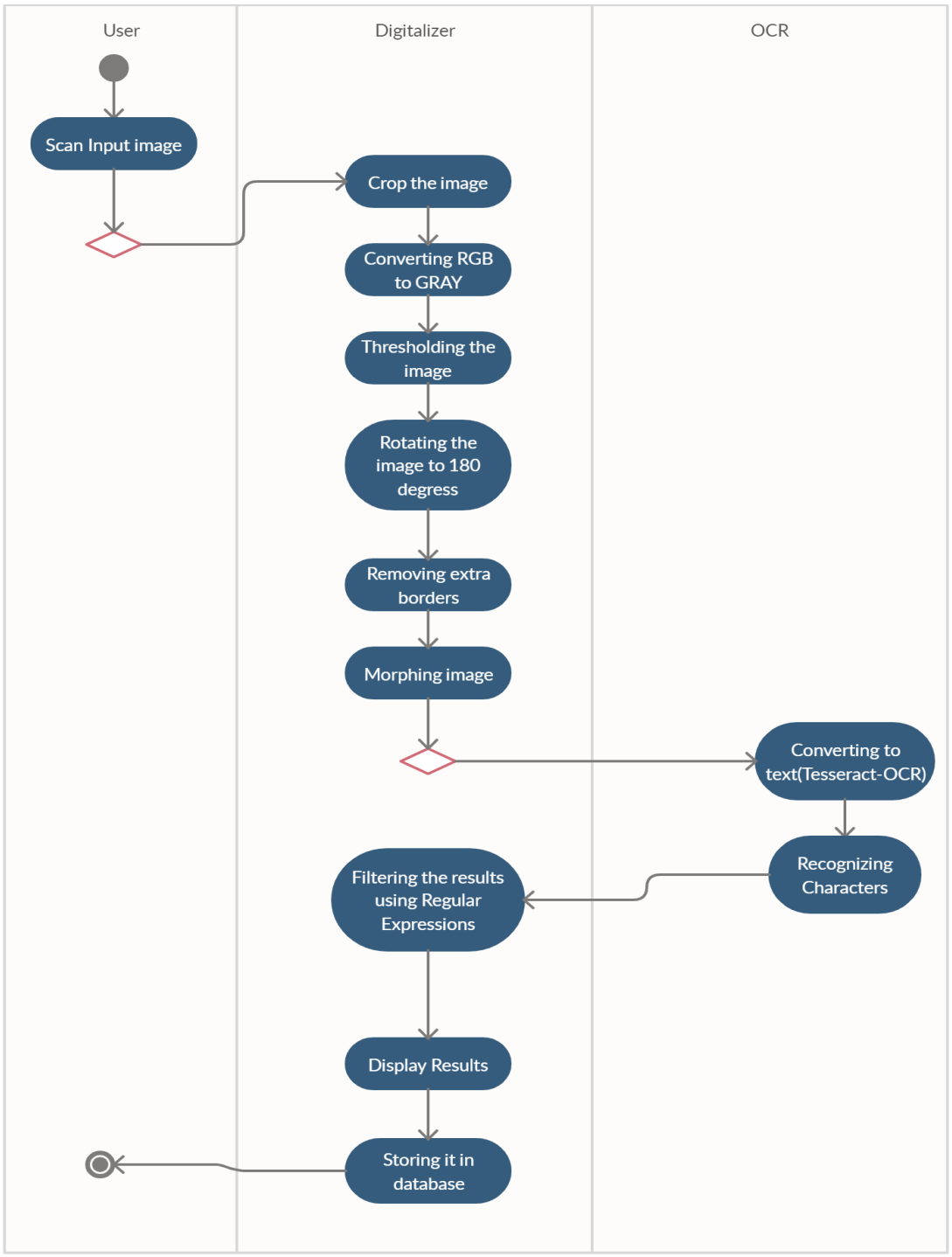

Figure 4.3 Activity Diagram for **CDEDI**

4.4 Sequence Diagram

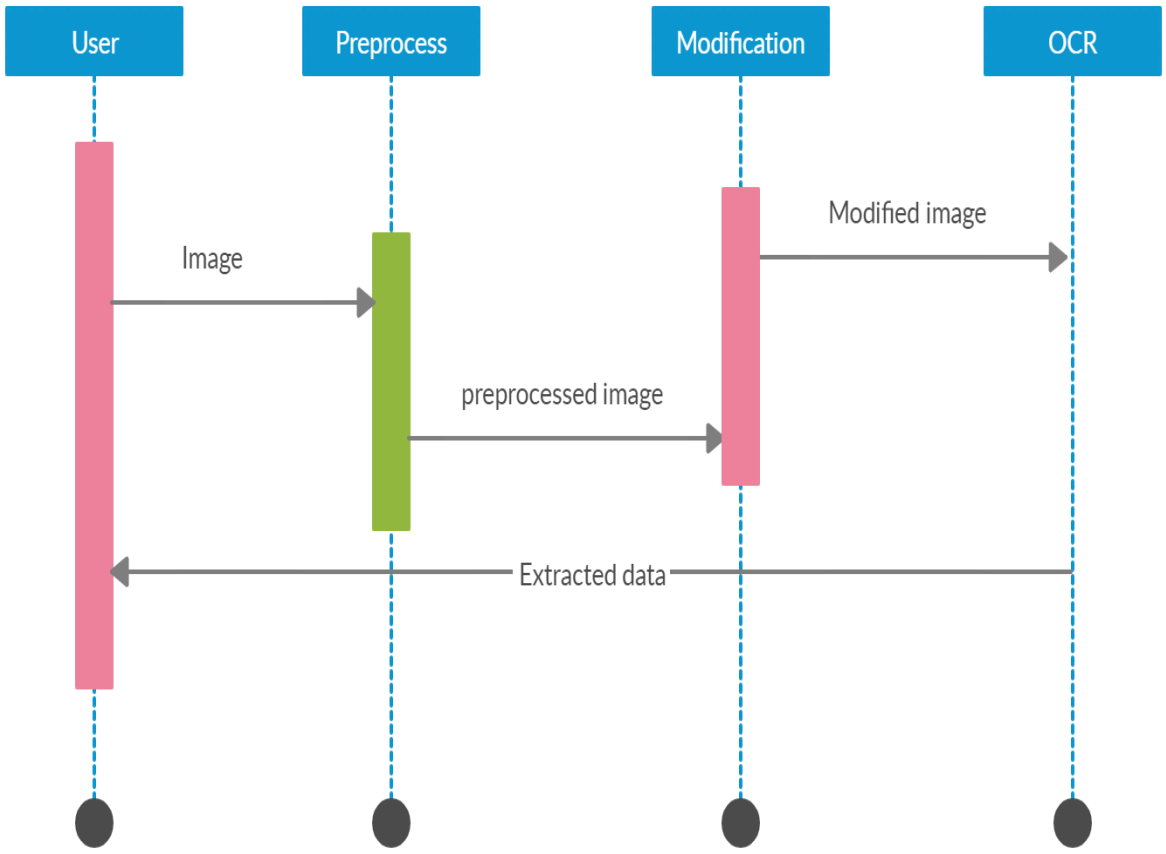

Figure 4.4 Sequence Diagram for **CDEDI**

4.5 System Architecture

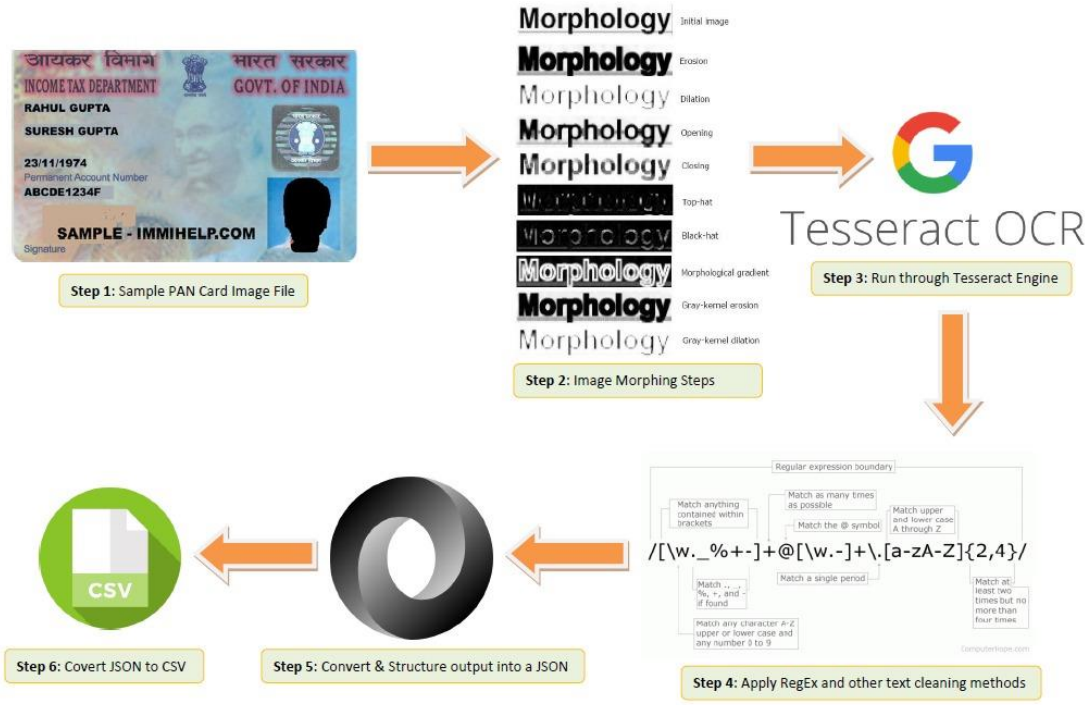

Figure 4.5 System Architecture - **CDEDI**

The important phases of optical character recognition include pre-processing, segmentation, normalization, feature extraction, classification and post processing. For designing an application with high efficiency it is important to understand the difficulties that may advance in each phase.

Pre-processing Phase:

The aim of pre-processing is to eliminate unbidden characteristics in an image without omitting any crucial information. Preprocessing techniques are performed on color, grey-level or binary document images containing text. It reduces the inconsistent

data and noise. Preprocessing enhances the image, converts it into a suitable format and makes the image suitable for the next phases. This phase involves reduction of inconsistency, noise of the image which is a major issue and helps in efficient character recognition. Some of the techniques applied on the images in this phase are listed in the Table 4.1.

Table 4.1 Technique and Description

Technique	**Description**
Binarization	Separates image pixels as text or background
Noise reduction	Better improvements of image acquisition devices due to technology advancements
Skew Correction	Because of the possibility of rotation of the input image, document skew should be corrected
Morphological Operations	Adding or removing pixels to the characters that have holes or surplus pixels
Thresholding	Separation of information from image's background

Morphology:

Morphology is the method of processing digital images on the basis of shape. The options used in this project are defined in Table 4.2

Table 4.1 Technique and Description

Morphing option	**Description**
Dilate	Dilate is commonly known as "fill", "expand", or "grow". It can be used to fill "holes" of a size equal to or smaller than the structuring element
Erode	Erode does to the background what dilation does to the foreground. Given an image and a structuring element, erode can be used to remove islands smaller than the structuring element.

Segmentation Phase:

The crux of OCR is segmentation of text lines from the images. Segmentation is the process of segregating text from regions of interest, which are recognized characters. Document segmentation which is a major phase, involves the process of differentiating a document image into regions, in such a way that each region contains unique information. The accuracy of the system depends on accuracy of the page segmentation algorithm used [1].

There are majorly three categories of algorithms of document segmentation.

i. Top-down methods
ii. Bottom-up methods
iii. Hybrid methods

In top-down approach[3], the document segments are recursively divided into smaller regions and terminates when the criterion is met. The ranges obtained comprise the result of final segmentation.

In bottom-up approach, the pixels present in the region are searched and are grouped based on interest. Then these grouped pixels are turned into connection components that constitute characters which in turn combine into words and lines[2].

Hybrid approach is a combination of both top-down and bottom-up approaches.

The obtained isolated characters are minimized to a particular size based on the algorithm used. The segmentation phase is important as it converts the image in the form of m*n matrix. These matrices are the commonly normalized by minimizing the size and eliminating unnecessary information from the image without overlooking any significant information.

4.6 Technology Description

What is Python?

Python is widely used in the software development industry. There are many of reasons for this.

i) **High-level object-oriented programming language:** Python includes effective symbolism.
ii) **Rapid application development:** Because of its concise code and literal syntax, the development of applications gets accelerated. The reason for its wide usability is its simple and easy-to-master syntax. The simplicity of the code helps reduce the time and cost of development.
iii) **Dynamic typescript:** Python has high-level incorporated data structures blended with dynamic typescript and powerful binding.

Some of the unique features that make Python the most ubiquitous language among the developer community are:

a) Python supports code reusability and modularity.
b) It has a quick edit-inspect-debug cycle.
c) Debugging is straightforward in Python programs.
d) It has its own debugger written in Python itself, declaring to Python's reflective power.
e) Python includes a plethora of third-party components present in the Python Package Index (PyPI).

What is PyCharm?

PyCharm is a hybrid-platform developed by JetBrains as an IDE for Python. It is commonly used for Python application development. Some of the unicorn organizations such as Twitter, Facebook, Amazon, and Pinterest use PyCharm as their Python IDE!

It supports two versions: v2.x and v3.x.

We can run PyCharm on Windows, Linux, or Mac OS. Additionally, it contains modules and packages that help programmers develop software using Python in less time and with minimal effort. Further, it can also be customized according to the requirements of developers.

Features of PyCharm:

i) Intelligent Code Editor

ii) Code Navigation

iii) Refactoring

iv) Assistance for Many Other Web Technologies

v) Support for Popular Python Web Frameworks

vi) Assistance for Python Scientific Libraries

What is tesseract?

Tesseract — is an optical character recognition engine with open-source code, this is the most popular and qualitative OCR-library.OCR uses artificial intelligence for text search and its recognition on images.Tesseract is finding templates in pixels, letters, words and sentences. It uses two-step approach that calls adaptive recognition. It requires one data stage for character recognition, then the second stage to fulfill any letters, it wasn't insured in, by letters that can match the word or sentence context.

What is argparse?

The **argparse** module provides tools for writing very easy to use command line interfaces. It handles how to parse the arguments collected in sys.argv list, automatically generate help and issues error message when invalid options are given.

First step to desing the command line interface is to set up parser object. This is done by **ArgumentParser()** function in argparse module. The function can be given an explanatory string as description parameter.

What is csv?

CSV (Comma Separated Values) is a simple **file format** used to store tabular data, such as a spreadsheet or database. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.
For working CSV files in python, there is an inbuilt module called **csv**.

What is ftfy?

ftfy fixes Unicode that's broken in various ways. It works in Python 2.7, Python 3.2, or later. The goal of ftfy is to take in bad Unicode and output good Unicode, for use in your Unicode-aware code. This is different from taking in non-Unicode and outputting Unicode, which is not a goal of ftfy. It also isn't designed to protect you from having to write Unicode-aware code.

What is io?

The io module provides Python's main facilities for dealing with various types of I/O. There are three main types of I/O: *text I/O*, *binary I/O* and *raw I/O*. These are generic categories, and various backing stores can be used for each of them. A concrete object belonging to any of these categories is called a file object. Other common terms are *stream* and *file-like object*.

Independent of its category, each concrete stream object will also have various capabilities: it can be read-only, write-only, or read-write. It can also allow arbitrary random access (seeking forwards or backwards to any location), or only sequential access (for example in the case of a socket or pipe).

What is json?

The full-form of JSON is JavaScript Object Notation. It means that a script (executable) file which is made of text in a programming language, is used to store and transfer the data. Python supports JSON through a built-in package called json. To use this feature, we import the json package in Python script. The text in JSON is done through quoted string

which contains value in key-value mapping within { }. It is similar to the dictionary in Python.

What is numpy?

NumPy is a python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

Why use numpy?

In Python we have lists that serve the purpose of arrays, but they are slow to process.NumPy aims to provide an array object that is up to 50x faster that traditional Python lists.The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.Arrays are very frequently used in data science, where speed and resources are very important

What is 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.

What is os?

The OS module in python provides functions for interacting with the operating system. OS, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.

What is PIL?

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. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it. Pillow was announced as a replacement for PIL for future usage. Pillow supports a large number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.

This module is not preloaded with Python. So to install it execute the following command in the command-line:

pip install pillow

What is pytesseract?

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images.

Python-tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file

What is sys?

This **sys module** provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It provides information about constants, functions and methods of python interpreter. It can be used for manipulating Python runtime environment.

5. IMPLEMENTATION AND TESTING

5.1IMPLEMENTATION

Implementation has been divided into three parts.

i. Preprocessing the image

ii. Morphology and rotation

iii. Applying OCR to extract data

5.1.1 Preprocessing The Image

In this step the image is taken and initially converted into gray scale. The image's foreground and background are flipped in such a way that foreground is "white" and the background is "black". Thresholding of the image is done using cv2.threshold function where all the foreground pixels are set to 255 and background pixels are set to 0.The obtained co-ordinates of pixel values are then classified in such a way that co-ordinates greater than zero are considered and then bounding boxes are drawn in that region. Based on the orientation of the image, angle is computed. Then image rotation is done to deskew it. The co-ordinate pixels are then transformed into a 2d matrix using cv2.getRotationMatrix2D() function. The image then undergoes affine transformation, which is a geometrical transformation that preserves lines and parallelism. The correction angle is drawn on the image for validation using cv2.putText(). The processed image is then displayed as output in this step and is used as input for the next step.

Program:

```
import numpy as np
import argparse
import cv2

def align_image(img):
```

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

gray = cv2.bitwise_not(gray)

cv2.imshow("gray_img", gray)

Figure 5.1.1(1) gray image

thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
cv2.imshow("thresh",thresh)

Figure 5.1.1(2) image after intial threshold

cord = np.column_stack(np.where(thresh > 0))

angle = cv2.minAreaRect(cord)[-1]

if angle < -45:

angle = -(90 + angle)

```
    else:
        angle = -angle
    (height, width) = img.shape[:2]
    center = (width // 2, height // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotate = cv2.warpAffine(img, M,(width, height),flags=cv2.INTER_CUBIC,
borderMode=cv2.BORDER_REPLICATE)
    cv2.imshow("Input", img)
    cv2.imshow("Rotated", rotate)
```

Figure 5.1.1(3) image after rotation

```
   cv2.waitKey(0)
   cv2.destroyAllWindows()
    cv2.putText(rotate, "Angle: {:.2f} degrees".format(angle), (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    print("angle: {:.3f}".format(angle))
    processed_(rotate, img)

def processed_(x, y):
    cv2.imshow("Image", y)
    cv2.imshow("Rotated", x)
    cv2.imwrite("rotate.jpg", y)
```

Figure 5.1.1(4) image after final rotation

```
    cv2.waitKey(0)
    cv2.destroyAllWindows()

if __name__ == '__main__':
        ap = argparse.ArgumentParser()
    ap.add_argument("-i", "--image", required=True, help="path to input image file")
    args = vars(ap.parse_args())
    image_ = cv2.imread(args["image"])
            align_image(image_)
```

5.1.2 Morphology And Rotation

In this step, the image is morphed i.e., the structure element shape is modified and morphological operations are applied to the image to get desired results. Mathematical morphology is a method of processing digital images on the basis of shape. Morphing is available when working with images, surfaces, and contours. Here, we have used two morphological options: erode, dilate.

Dilate is commonly known as “fill”, or “grow”. It can be used to fill holes of a size equal to or smaller than the structuring element.

Erode does to the background what dilate does to the foreground. Given an image and a structuring element, erode can be used to remove islands smaller than the structuring element.

The structuring element used here is ellipse. In this step the morphological operation results are printed on the image for validation.

Program:

```
from __future__ import print_function
import cv2 as cv

def draw_str(dst, target, s):
    x, y = target
    cv.putText(dst, s, (x + 1, y + 1), cv.FONT_HERSHEY_PLAIN, 1.0, (0, 0, 0), thickness=2, lineType=cv.LINE_AA)
    cv.putText(dst, s, (x, y), cv.FONT_HERSHEY_PLAIN, 1.0, (255, 255, 255), lineType=cv.LINE_AA)

def modify(img, modes, str_modes):
    sz = 10
    itn = 1
    opern = modes.split('/')
    sz = sz - 10
    op = opern[sz > 0]
    sz = sz * 2 + 1

    str_name = 'MORPH_' + str_modes.upper()
    operation_name = 'MORPH_' + op.upper()
    st = cv.getStructuringElement(getattr(cv, str_name), (sz, sz))
    res = cv.morphologyEx(img, getattr(cv, operation_name), st, iterations=itn)

    draw_str(res, (10, 20), 'mode: ' + modes)
    draw_str(res, (10, 40), 'operation: ' + operation_name)
    draw_str(res, (10, 60), 'structure: ' + str_name)
```

```
  draw_str(res, (10, 80), 'ksize: %d  iterations: %d' % (sz, itn))
    cv.imshow('morphology', res)
    cv.imwrite('result.jpg', img)

if __name__ == '__main__':

    import sys

    try:
        fn = sys.argv[1]
    except:
        fn = '16.jpg'
    img_ = cv.imread(fn)
    if img_ is None:
        print('Failed to load image file:', fn)
        sys.exit(1)
    modes_ = 'erode/dilate'
    str_modes_ = 'ellipse'

    cv.namedWindow('morphology')
    modify(img_, modes_, str_modes_)
    cv.waitKey(0)
    cv.destroyAllWindows()
```

Figure 5.1.2(1) image after morphology

5.1.3 Applying OCR To Extract Data:

The image is taken and is converted into gray scale. Then it is checked from the command line arguments to apply thresholding to preprocess the image. Then we should check if blurring is to be done to remove the noise. The gray scale image is then written to the disk as temporary file to apply ocr on it. Then we load the image as PIL/Pillow image on which ocr is applied and the file is then removed. The data from the image is extracted using pytesseract.image_to_string().The extracted data is then written to a text file named "outputbase.txt". Then using ftfy.fix_text(), fix_encoding() removal of gibberish text is done. The data is then classified accordingly and unfavorable characters are replaced to get desired results. After extraction of all variables, a dictionary is created and then the data is written to JSON file for further use.

Program:

```
from PIL import Image
import pytesseract
import argparse
import cv2
import os
import re
import io
import json
import ftfy
import csv
from numpy import unicode

pytesseract.pytesseract.tesseract_cmd=r"C:\Users\HOME\AppData\Local\Tessera
ct-OCR\tesseract.exe"


def ocr_apply(args):

    image = cv2.imread(args["image"])
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    if args["preprocess"] == "thresh":
        gray = cv2.threshold(gray, 0, 255,
                             cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
```

```
elif args["preprocess"] == "adaptive":
    gray = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 31, 2)

if args["preprocess"] == "linear":
    gray = cv2.resize(gray, None, fx=2, fy=2, interpolation=cv2.INTER_LINEAR)

elif args["preprocess"] == "cubic":
    gray = cv2.resize(gray, None, fx=2, fy=2, interpolation=cv2.INTER_CUBIC)


if args["preprocess"] == "blur":
    gray = cv2.medianBlur(gray, 3)

elif args["preprocess"] == "bilateral":
    gray = cv2.bilateralFilter(gray, 9, 75, 75)

elif args["preprocess"] == "gauss":
    gray = cv2.GaussianBlur(gray, (5, 5), 0)
cv2.imshow('Proccesed', gray)
cv2.waitKey(0)
cv2.destroyAllWindows()

        filename = "{}.png".format(os.getpid())
        cv2.imwrite(filename, gray)

text = pytesseract.image_to_string(Image.open(filename), lang='eng')
os.remove(filename)


text_output = open('outputbase.txt', 'w', encoding='utf-8')
text_output.write(text)
text_output.close()

file = open('outputbase.txt', 'r', encoding='utf-8')
text = file.read()

text = ftfy.fix_text(text)
text = ftfy.fix_encoding(text)


name = None
fname = None
dob = None
pan = None
nameline = []
dobline = []
panline = []
```

```
text0 = []
text1 = []
text2 = []


lines = text.split('\n')
for lin in lines:
    s = lin.strip()
    s = lin.replace('\n', '')
    s = s.rstrip()
    s = s.lstrip()
    text1.append(s)

text1 = list(filter(None, text1))

lineno = 0

for wordline in text1:
    xx = wordline.split('\n')
    if ([w for w in xx if re.search(
            '(INCOMETAXDEPARWENT @|INCOME TAX DEPARTMENT|INCOME TAX DEPARTMENT ¢|INCOMETAX DEPARTMENT gs GOV\'T OF Be|mcommx|INCOME|TAX|GOW|GOVT|GOVERNMENT|OVERNMENT|VERNMENT|DEPARTMENT|EPARTMENT|PARTMENT|ARTMENT|INDIA|NDIA)$',
            w)]):
        text1 = list(text1)
        lineno = text1.index(wordline)
        break


text0 = text1[lineno + 1:]


def findword(textlist, wordstring):
    lineno = -1
    for wordline in textlist:
        xx = wordline.split()
        if [w for w in xx if re.search(wordstring, w)]:
            lineno = textlist.index(wordline)
            textlist = textlist[lineno + 1:]
            return textlist
    return textlist

remove_lower = lambda text: re.sub('[a-z]', '', text)
try:

    name = text0[0]
    name = name.strip()
    name = name.replace("8", "B")
```

```
name = name.replace("0", "D")
name = name.replace("6", "G")
name = name.replace("1", "I")
name = name.replace("\'", "")
name = re.sub('[^A-Z] +', ' ', name)
result = remove_lower(name)
name = result
name = name.strip()

fname = text0[1]
fname = fname.strip()
fname = fname.replace("8", "S")
fname = fname.replace("0", "O")
fname = fname.replace("6", "G")
fname = fname.replace("1", "I")
fname = fname.replace("\"", "A")
fname = fname.replace("\'", "")
fname = re.sub('[^A-Z] +', ' ', fname)
result = remove_lower(fname)
fname = result
fname = fname.strip()

# Cleaning DOB
dob = text0[2]
dob = dob.rstrip()
dob = dob.lstrip()
dob = dob.replace('l', '/')
dob = dob.replace('L', '/')
dob = dob.replace('I', '/')
dob = dob.replace('i', '/')
dob = dob.replace('|', '/')
dob = dob.replace('\'', '/1')

remove_extra = lambda text: re.sub('[^0-9/]', '', text)
dob = dob.lstrip()
dob = dob.rstrip()
result = remove_extra(dob)
dob = result

text0=findword(text1,
   '(Pormanam|Number|umber|Account|ccount|count|Permanent|ermanent|m
   anent|wumm)$')
panline = text0[0]
pan = panline.rstrip()
pan = pan.lstrip()
pan = pan.replace(" ", "")
pan = pan.replace("\"", "")
pan = pan.replace(";", "")
pan = pan.replace("%", "L")
remove_other = lambda text: re.sub('[^A-Z0-9]', '', text)
```

```
        result = remove_other(pan)
        if len(result) > 10:
            pan = result[:10]
        else:
            pan = result
    except:
        pass
    data = {'Name': name, 'Father Name': fname, 'Date of Birth': dob, 'PAN': pan}
            print(data)
try:
    to_unicode = unicode
except NameError:
    to_unicode = str

with io.open('data.json', 'w', encoding='utf-8') as outfile:
    str_ = json.dumps(data, indent=4, sort_keys=True, separators=(',', ': '),
ensure_ascii=False)
    outfile.write(to_unicode(str_))

with open('data.json') as json_file:
    data = json.load(json_file)

employee_data = data

fields = employee_data
l = list()
l.append(fields)
try:
    with open('data_file.csv', 'r') as csvfile:
        count = 1
except:
    count = 0
with open('data_file.csv', 'a') as csvfile:

    writer = csv.DictWriter(csvfile, fieldnames=fields)

    if count == 0:
        writer.writeheader()

    writer.writerows(l)

if __name__ == '__main__':
    # construct the argument parse and parse the arguments
    ap = argparse.ArgumentParser()
    ap.add_argument("-i", "--image", required=True,
                help="path to input image to be OCR'd")
    ap.add_argument("-p", "--preprocess", type=str, default="thresh",
help="type of preprocessing to be done, choose from blur, linear, cubic or
bilateral")
    args1 = vars(ap.parse_args())
```

ocr_apply(args1)

Output:

Adaptive Thresholding:

Figure 5.1.3(1) final image with adaptive thresholding

```
(venv) C:\Users\USER\PycharmProjects\mini1>python final_ocr.py -i 3.jpg  -p adaptive
{'Name': 'P  9.', 'Father Name': 'I ,', 'Date of Birth': None, 'PAN': None}
```

Figure 5.1.3(2) output with adaptive thresholding

Bilateral Filtering:

Figure 5.1.3(3) final image with bilateral filtering

```
(venv) C:\Users\USER\PycharmProjects\mini1>python final_ocr.py -i 3.jpg  -p bilateral
{'Name': '_', 'Father Name': 'ANYPANAYOU', 'Date of Birth': '/', 'PAN': 'NYOT'}
```

Figure 5.1.3(4) output with bilateral filtering

Blur (Median blur):

Figure 5.1.3(5) final image with median blur

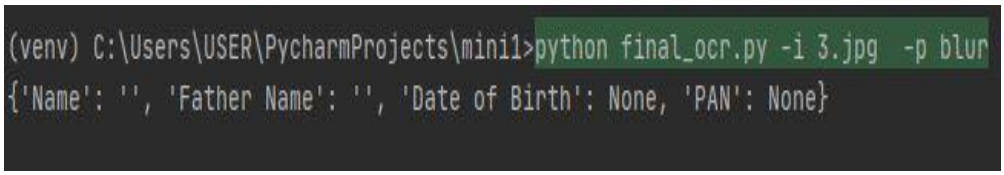

Figure 5.1.3(6) output with median blur

Cubic Interploation:

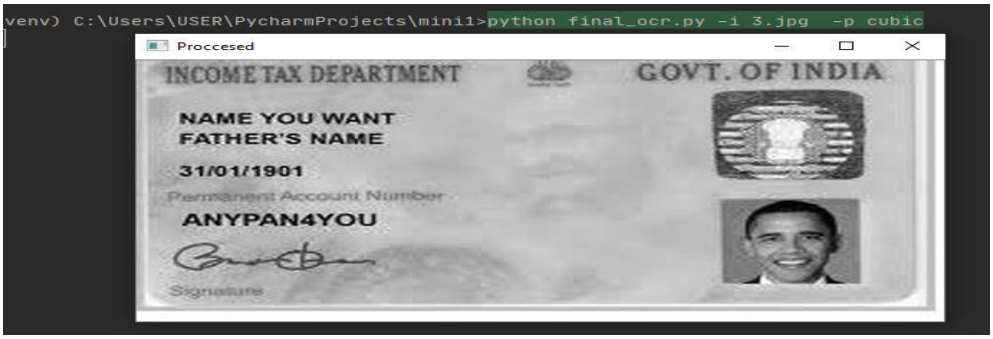

Figure 5.1.3(7) final image with cubic interpolation

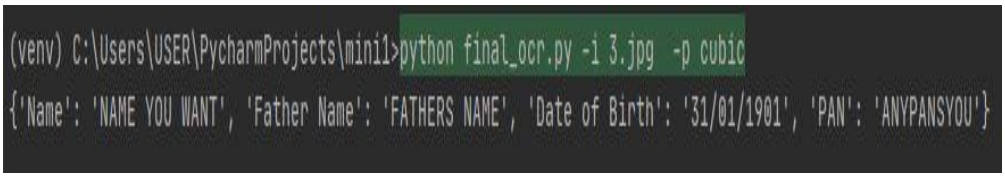

Figure 5.1.3(8) output with cubic interpolation

Gaussian Blur:

Figure 5.1.3(9) final image with gaussian blur

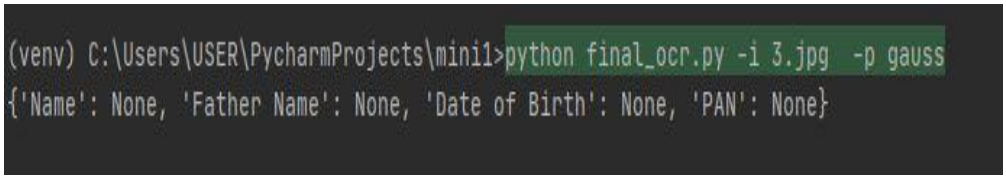

Figure 5.1.3(10) output with gaussian blur

Linear Interpolation:

Figure 5.1.3(11) final image with linear interpolation

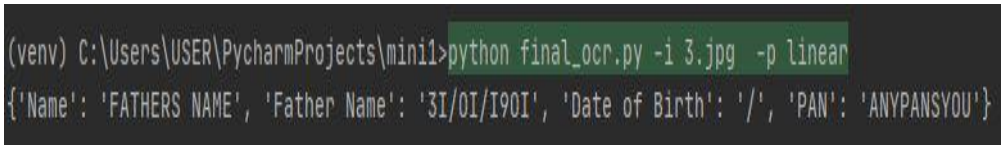

Figure 5.1.3(12) output with linear interpolation

Threshold:

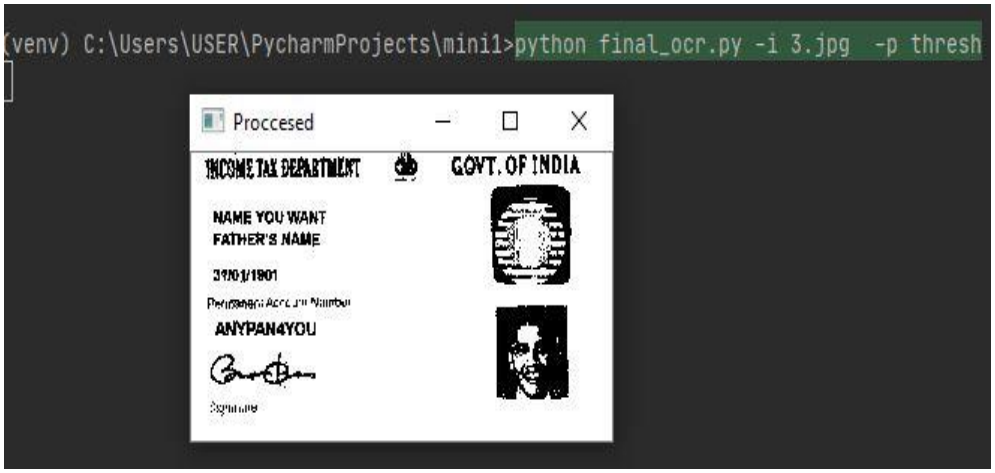

Figure 5.1.3(13) final image with final thresholding

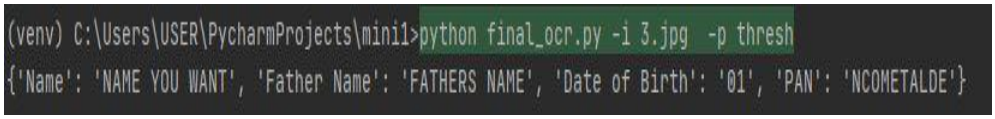

Figure 5.1.3(14) output with final thresholding

Table 5.1 Data stored in EXCEL Sheet

S.No.	Date Of Birth	Father Name	Name	PAN
1	6/12/1979	RAMAKRISHNA SARMA ADDALA	SRINIVAS ADDALA	AIHPA8488P
2	30/11/1973	KESAV PRASAD GUPTA	AMILA SHAH	FISPS0735P
3	31/01/1901	FATHERS NAME	NAME YOU WANT	ANYPANSYOU
4	1/0/1980	CCHANDRAKANT JAYANTILAL SORT	SONI MINARIBENS	SITRATTART
5	1/2/1985	DEBARAJA SAHU AG	SANTOSH JUMAR SAHU Z Â¥	DCMPS9585P
6	15/05/1987	SHANTILAL PANCHAL	HIMANSHU PANCHAL	AWWIP1078K
7	5/1/1981	BRINDABON MONDAL	CHADONA MONDAL	AWSPM23546
8	2/6/1976	PRAKASH MEHENDALE	ADITYA MEHENDALE	BODPM4264E
9	10/12/1983	CHHOTU RAM	AMAR SINGH	BLUPS4233F
10	3/5/1988	NABADIN MIAN	ALAUDDIN MIAN	BFDPM6385P
11	10/6/1985	TEKURI VENKATESU	TEKURI NARASIMHULU	AHVNPNO716E
12	01/01/1990	BR MISHRA	RAJESH BALKRISHNA MISHRA P	AUUPME954D
13	25/08/1990	SANJEEV CHOPRA	PRACHIT CHOPRA GF F	AMWPC3594M
14	16/07/1986	DURAISAMY	D MANIKANDAN	BNZPM2501F

15	15/10/1994	AWADHESH KUMAR PANDEY	MANOJ KUMAR PANDEY	INCOMETAXD

5.2TESTING:

Input image:

Figure 5.2(1) input image

Gray-scale image:

Figure 5.2(2) gray image

Thresholded Image:

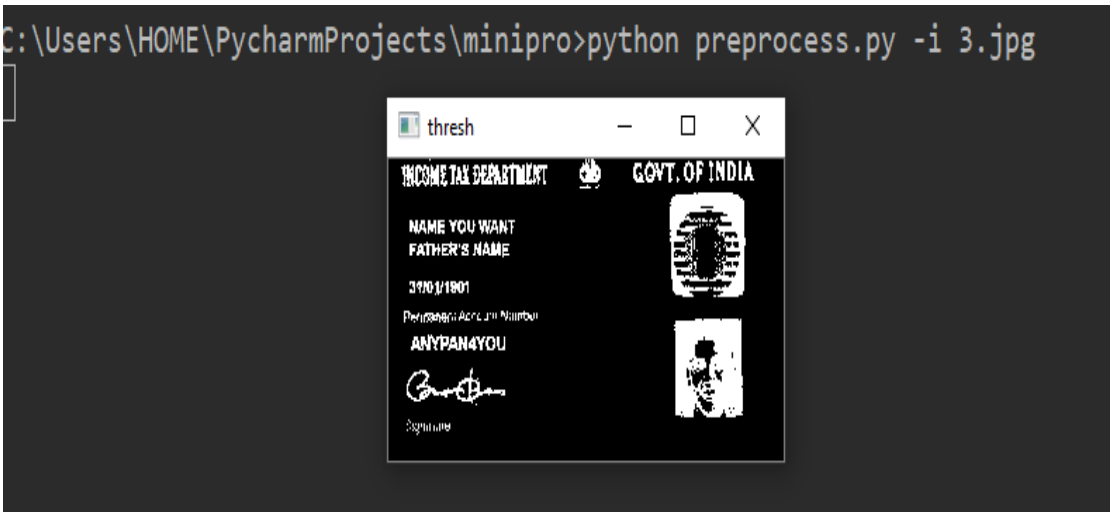

Figure 5.2(3) intial thresholded image

Rotated image:

Figure 5.2(4) rotated image

Image after performing morphology:

Figure 5.2(5) image after morphology

Processed Image:

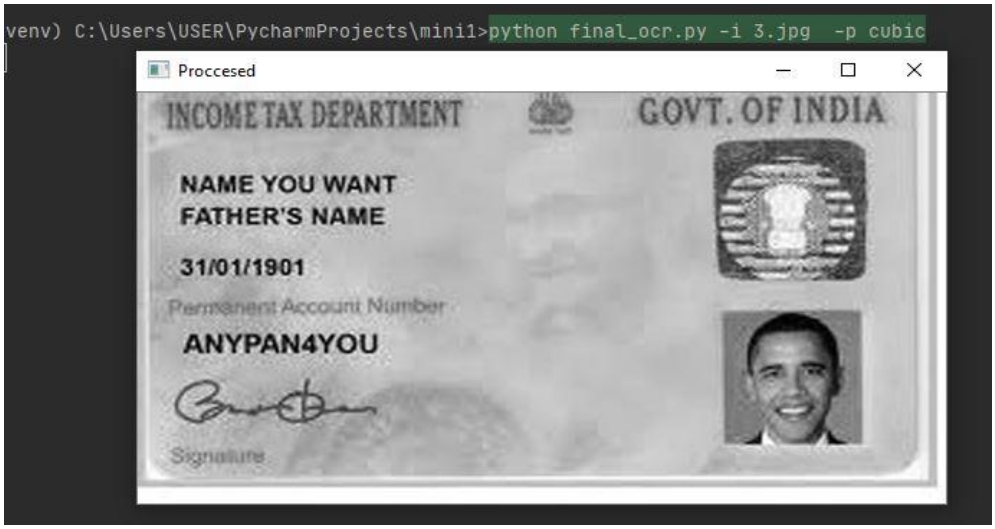

Figure 5.2(6) processed image

Output:

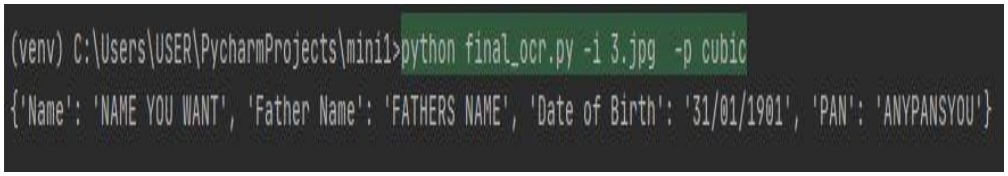

Figure 5.2(7) final output

Data.json file:

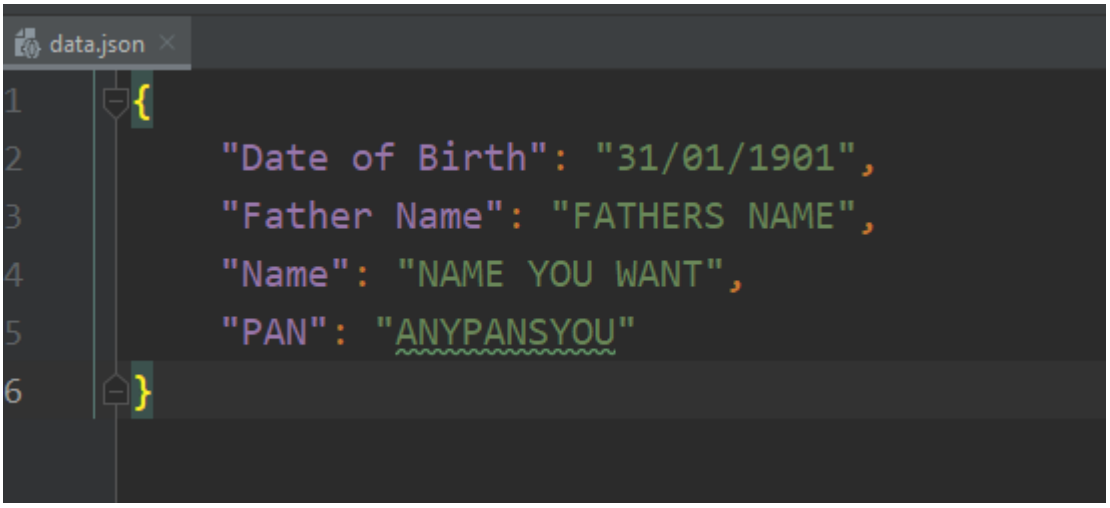

Figure 5.2(8) result json file

Accuracy:

The accuracy of results obtained:

Table 5.2 Accuracy Data stored in EXCEL Sheet

S.NO	Image name	Date of Birth (%)	Father's Name (%)	Name (%)	PAN (%)
1	Image 1	90	100	100	100
2	Image 2	100	100	100	100
3	Image 3	100	100	100	90
4	Image 4	80	70	80	0
5	Image 5	80	80	60	100
6	Image 6	100	100	100	70
7	Image 7	80	100	100	90
8	Image 8	80	100	100	100
9	Image 9	100	100	100	100
10	Image 10	80	100	100	100
11	Image 11	90	100	100	90
12	Image 12	100	100	90	100
13	Image 13	100	100	70	100
14	Image 14	100	100	100	100
15	Image 15	100	100	100	0

6. CONCLUSION & FUTURE SCOPE

Numerous algorithms, methods and techniques have been proposed to optical character recognition in scene imagery, yet there are not enough literature surveys in this field. In this project, we have attempted to obtain maximum accuracy using pre-defined models and techniques. Firstly, we introduced optical character recognition system, discussed major challenges of OCR, then we discussed in great detail the main important phases, architecture, proposed system for digitalizing the data for PAN cards as an example. Finally the implementation of the code and testing is discussed. Although the state-of-the art OCR enables text recognition with high accuracy, we think that there could be many more practical applications of OCR.As a future work, we are planning to incorporate mobile devices with OCR in one CR system. An automated book reader or a receipt tracker constitutes some of our future OCR based applications.

REFERENCES

[1] S. Jaeger, S. Manke, J. Reichert, and A. Waibel, "Online handwriting recognition: the npen++ recognizer," International Journal on Document Analysis and Recognition, vol. 3, no. 3, pp. 169–180, 2001

[2] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in Advances in neural information processing systems, pp.545–552,2009.

[3] Kyong-Ho Lee, Yoon-Chul Choy and Sung-Bae Cho **,"**Geometric Structure Analysis Of Document Images: A Knowledge-Based Approach" ,2000.

[4] Mehak Naz Mangoli, Prof.Sujata Desai "Optical Character Recognition for Cursive Handwriting "Department of Computer Science and Engineering, BLDEA College of Engineering and Technology, Karnataka, 2016.

[5] https://www.researchgate.net/publication/311851325_A_Detailed_Analysis_of_Optical_Character_Recognition_Technology

[6] https://github.com/tesseract-ocr/docs/blob/master/tesseracticdar2007.pdf

[7] https://nanonets.com/blog/ocr-with-tesseract/