

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering
Degree in Computer Science and Engineering

By

NAME: BODAPATI SOHAN CHIDVILAS
(Reg. No.38110688)
NAME: VENKATA NAGA SAI RAKESH KAMISSETTY
(Reg. No.38110635)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL
OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY JEPPIAAR
NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

MAY 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)



Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119
www.sathyabamauniversity.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **BODAPATI SOHAN CHIDVILAS (38110688), VENKATA NAGA SAI RAKESH KAMISSETTY (38110635)** who carried out the project entitled “**DIGITIZATION OF DATA FROM INVOICE USING OCR**” under my supervision from August 2021 to November 2021.

Internal Guide

Name: Dr. S. Revathy M.E., Ph.D.,

Head of the Department

Name: Dr. S. VIGNESHWARI M.E., Ph.D.,

Name: Dr. L. LAKSHMANAN M.E., Ph.D.,

Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

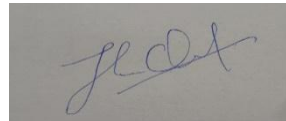
I **BODAPATI SOHAN CHIDVILAS** and **VENKATA NAGA SAI RAKESH KAMISETTY** hereby declare that the Project Report entitled **DIGITIZATION OF DATA FROM INVOICE USING OCR** done by me under the guidance of **Dr. S.Revathy M.E., Ph.D.**, (Internal) at **cSoft Technologies** (Company name and address) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 09-11-2021

**BODAPTI SOHAN CHIDVILAS
VENKATA NAGA SAI RAKESH KAMISETTY**

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

A rectangular box containing a handwritten signature in blue ink. The signature is cursive and appears to read 'Bodapati Sohan Chidvilas'.

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing , **Dr.S.Vigneshwari M.E., Ph.D., and Dr.L.Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S.Revathy M.E., Ph.D.,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



Date: Sep 10th 2021

Bangalore, India

To Whom It May Concern:

Subject: Completion of Internship

This is to certify that Mr. Bodapati Sohan Chidvilas from Sathyabama Institute of Science and Technology, Chennai completed his internship from July 1st to Aug 30th, 2021 and helped us advance our knowledge of machine learning in extracting text from images.

During this course of the internship, **Mr. Sohan** worked on building a model that:

1. that helps extracting data in JSON format from scanned invoice images with data accuracy >80%.
2. Used open source libraries to achieve the goal.

Mr. Sohan primarily used Python to develop his working code. He was able to work independently and progress on his given goals satisfactorily.

Should you have any questions on his work and his output, please do not hesitate to contact me.

Thank you,

Chiamala Aravamudhan

CEO

cSoft Technologies Pvt. Ltd.

admin@csoft-tech.com

Regd. Office: F1, New No.10, East Mada Street, Velachery, Chennai TN 600042 IN

TRAINING CERTIFICATE

cSoft Technologies

Developing Innovative Solutions

Date: Sep 10th 2021

Bangalore, India

To Whom It May Concern:

Subject: Completion of Internship

This is to certify that Mr. Rakesh K from Sathyabama Institute of Science and Technology, Chennai completed his internship from July 1st to Aug 30th, 2021 and helped us advance our knowledge of machine learning in extracting text from images.

During this course of the internship, **Mr. Rakesh** worked on building a model that:

1. that helps extracting data in JSON format from scanned invoice images with data accuracy >80%.
2. Used open source libraries to achieve the goal.

Mr. Rakesh primarily used Python to develop his working code. He was able to work independently and progress on his given goals satisfactorily.

Should you have any questions on his work and his output, please do not hesitate to contact me.

Thank you,



Chiamala Aravamudhan

CEO

cSoft Technologies Pvt. Ltd.

admin@csoft-tech.com

ABSTRACT

Optical Character Recognition (OCR) is a predominant aspect to transmute scanned images and other visuals into text. Computer vision technology is extrapolated onto the system to enhance the text inside the digitized image. This preliminary provisional setup holds the invoice's information and converts it into JSON and CSV configurations. This model can be helpful in divination based on knowledge engineering and qualitative analysis in the nearing future. The existing system contains data extraction and nothing more. In a paramount manner, image pre-processing techniques like black and white, inverted, noise removal, grayscale, thick font, and canny are applied to escalate the quality of the picture. With the enhanced image, more OpenCV procedures are carried through. In the very next step, three different OCRs are used: Keras OCR, Easy OCR, and Tesseract OCR, out of which Tesseract OCR gives the precise result. After the initial steps, the undesirable symbols (/t, /n) are cleared to get the escalated text as an output. Eventually, a unique work that is highly accurate in giving JSON and CSV formats is developed.

Impact statement— In our protrude, a front-end android app is developed which takes input from the user and stores the output onto the database. The JSON and CSV files can be viewed through an app by the end.

ABBREVIATIONS

OCR – Optical Character Recognition
JSON - JavaScript Object Notation
CSV – Comma-separated values

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No
	ABSTRACT	7
	LIST OF ABBREVIATIONS	7
1.	INTRODUCTION	9
	1.1 Project Statement	10
	1.2 Project Justification	10
2.	LITERATURE SURVEY	11
3.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	22
4.	EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED	23
	4.1 Language used	23
	4.2 Python libraries used	23
	4.3 Phases of work	
5.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	33
	5.1 Experimental results compared with in-use OCR results	33
	5.2 Subjective Results	33
	5.3 Limitations	34
6.	SUMMARY AND CONCLUSIONS	35
	REFERENCES	36
	APPENDIX	38

CHAPTER 1

INTRODUCTION

Computer vision drew attention by swaying as a data- reliant stratified feature in extraction methods.

Visualization technology has been imposed to decipher an image to make the machine understand. Optical Character Recognition (OCR) automatically extracts characters from the image and recognizes text quickly using an existential database. OCR is a meticulous technology that comes up with legible recognition of inscribed or in-written characters from images which will be further digitized in our apparatus. Various procedures have been in use already. Despite this, the existing OCRs cannot convert the text into the desired form that the end-user needs. In this current era, OCR has been the most dominant technology. OCR can be used in an enchanting number of ways apart from just extracting the text. They are shown in a different dimension here.

Among the OCRs around globe, the least preferred is Keras OCR as it goes with line segmentation. The other one is Easy OCR, a parasite of spaces. Finally, Tesseract OCR is the best open-source choice as it can be corelated with python libraries called pytesseract. Tesseract OCR extracts the text based on the invoice format. The exact explanation of how tesseract OCR extracts the text form the image is inscribed in section V under phase 4. The primary Python library used in our structure is OpenCV, which helps the machine find objects in the image and make OCR work efficiently. In this adorable framework, pdf or an image (JPG, JPEG, PNG) is taken as input from the android app. If it is a pdf, it will be converted into a picture, then pre-processing techniques (Black and white, no noise, Grayscale, thick font) have been used to amplify the text. Textual content is a conduit where details are confronted with a machine orderly to give a valid result. Multiple approaches are there to extract text in many different ways with an OCR to get the most accurate result. It has to be validated with a set of pre-trained images to get an efficient output. Then the noise should be filtered from the photo to make the above statement work. Below are the few processing methodologies for an image to be intensified under OpenCV.

Thresholding is a form where the image will be segmented to understand the image better. Several procedures have been applied like spatial to correspond with the pixels and further computerize it to black and white for highlighting the words to bring

back the highest quality. The pivot OpenCV methodology also sharpens the image by blurring the borders to make the essential fields stand out. Threshing also includes smoothening where it evens rough side to blend the text. The text got from the OCR may not be error-free. So, regular expressions have been used to clean the printed characters further. The string format must be converted into a list by splitting it as a ratio. In the concluded part of our setup, the cleaned text is converted into JSON and CSV formats for better comprehension.

Here JavaScript Object Notation (JSON) is a format that will return the object from the back-end server and edit cookies. Apart from the primary use, the web developers mostly use it to deploy output onto the web page. Mainly, Key pair values are generated and commonly known as dictionaries in python. CSV is a format where a comma separates the values, and a tabular column is created, which returns as an excel sheet. The basic idea of developing an app is to make it uncomplicated. A rudimentary java file picker has been evolved to residue the complexity.

1.1 PROBLEM STATEMENT

It is observed that the performance of the computer vision degrades drastically under the course of action. The results are also compared with existing computer vision fooling approaches to evaluate the accuracy drop. We propose a primary state-of-the-art performance using the solution in terms of the computer robustness under OCR is observed in the experiments. No OCR in the world which can detect only the specific contents with more than 80 percent accuracy which can convert it into JSON or CSV.

1.2 PROJECT JUSTIFICATION

The necessity of this protrude is extracting the relevant data instead of unnecessary matter. For instance, take medical bills, when we need only the tabular contents then there is no OCR that can detect the tabular columns separately, that to with 80 % accuracy and returning the output as JSON as well as csv. The main advantage of JSON and CSV is that the end user need not enter contents as they will directly return the particulars. Our methodology proves to be highly accurate while tested on a variety of input images of bills and invoices. This course of action achieves an increase in accuracy by 80%. The proposed approach can be used to improve the robustness of Computer Vision.

CHAPTER 2

LITERATURE SURVEY

2.1 OCR text extraction

Abstract:

This research tries to find out a methodology through which any data from the daily-use printed bills and invoices can be extracted. The data from these bills or invoices can be used extensively later on - such as machine learning or statistical analysis. This research focuses on extraction of final bill-amount, itinerary, date and similar data from bills and invoices as they encapsulate an ample amount of information about the users purchases, likes or dislikes etc. Optical Character Recognition (OCR) technology is a system that provides a full alphanumeric recognition of printed or handwritten characters from images. Initially, OpenCV has been used to detect the bill or invoice from the image and filter out the unnecessary noise from the image. Then intermediate image is passed for further processing using Tesseract OCR engine, which is an optical character recognition engine. Tesseract intends to apply Text Segmentation in order to extract written text in various fonts and languages. Our methodology proves to be highly accurate while tested on a variety of input images of bills and invoices.

2.2 Mixed-Initiative Approach to Extract Data from Pictures of Medical Invoice

Abstract:

Extracting data from pictures of medical records is a common task in the insurance industry as the patients often send their medical invoices taken by smartphone cameras. However, the overall process is still challenging to be fully automated because of low image quality and variation of templates that exist in the status quo. In this paper, we propose a mixed-initiative pipeline for extracting data from pictures of medical invoices, where deep-learning-based automatic prediction models and task-specific heuristics work together under the mediation of a user. In the user study with 12 participants, we confirmed our mixed-initiative approach can supplement the drawbacks of a fully automated approach within an acceptable completion time. We further discuss the findings, limitations, and future works for designing a mixed-initiative system to extract data from pictures of a complicated table.

2.3 OCR for Data Retrieval: An analysis and Machine Learning Application model for NGO social volunteering

Abstract:

With the increase in amount of information being made available in digital format, information retrieval is a challenging task. Currently there exists a gap between organizations, volunteers and NGOs for volunteering work. There has been an upsurge of NGOs, non-profit events and corresponding independent volunteers or organizations willing to interconnect especially during these pandemic times. There is a need to fill this gap and connect the stakeholders minimizing the emergency response times. This paper proposes a novel design and implementation of an OCR based application for Automated NGO connect using machine learning. Phases implemented include image de-noising, binarization, data extraction and data conversion. The framework integrates deep learning-based Tesseract OCR with image processing module and Data visualization module. The proposed model can be extended to other application domains as well for research purposes.

2.4 Comparative Analysis of Text Extraction from Color Images using Tesseract and OpenCV

Abstract:

Image-based Text Extraction has a growing requirement in today's generation. Students, doctors, and engineers generate a lot of images every day. It is very important to extract text from these images in a simple yet effective manner. We can obtain useful information by testing these images. We aim is to summarize the visual information and retrieve its content. The Optical Recognition System involves several algorithms that fulfill this purpose. Text Extraction involves a lot of processes from text detection, localization, segmentation and, text recognition. Tesseract is the most optimized OCR Engine build by HP Labs and owned by Google. Text Detection involves the recognition of text from desired input images. Text Localization involves identifying the position of text on the images. Tesseract works pretty well on the light-colored background but unable to recognize text on darker shades. We have tried to apply various image processing techniques. This method will allow us to recognize text from most types of background. We propose to provide methods for

easy text extraction. Track bar allows the user to adjust various parameters to extract a required text from an Image. This method is gaining huge importance in years to come. For Automation, we can use a set of image processing techniques such as edge detection, filtering and, blurring for better results. A series of these steps will enable us to extract text from images efficiently. This experiment compares the optimized result by two methods for efficient Text Extraction.

2.5 Analysis of Image Classification for Text Extraction from Bills and Invoices

Abstract:

Optical Character Recognition (OCR) technology offers a complete alphanumeric recognition of printed or handwritten characters from pictures such as scanned bills and invoices. Intelligent extraction and storage of text in structured document serves document analytics. The current research attempts to find a methodology through which any information from the printed invoice can be extricated. The intermediate image is passed over using an OCR engine for further processing. Segmentation extracts written text in various fonts and languages. Image classification helps in making a decision based on the classification results. This paper surveys these techniques and compares them in terms of metrics, algorithm and results.

2.6 Text Orientation Detection Based on Multi Neural Network

Abstract:

Optical character recognition (OCR) is an important research area in the field of pattern recognition, such as Vehicle License Plate Recognition. With it, we can extract textual information from the images to facilitate digital processing. However, most existing systems are designed to detect or recognize horizontal (or near-horizontal) texts and can't be applied to recognize texts of varying orientations. Text Orientation Detection is an important but challenging task. It can be used as a pre-processing for previous researches, and allows them to be adapted to more complex situations. Once we get angle of the text, we can use a series of transformations to get horizontal text. Although this problem is a multiclass classification, the results using common multiclass classification methods are not ideal. Our algorithm is inspired by the human behavior of recognizing texts. In this paper, we propose a new algorithm containing three neural networks to detect text orientation. The first is for capturing the abstract information of text image, and trained on multi-orientation, synthetic texts. Then the second is for evaluating the correctness of meaning of texts and trained on horizontal texts. The output of these two neural networks serves as the input to the last. In this way, the last neural network can obtain information about the image of the text as well as its meaning,

and finally evaluate and output the angle of the text. According to the results, our proposed method can guarantee a high accuracy of text orientation detection.

2.7 Data Extraction from Invoices Using Computer Vision

Abstract:

Management of invoices and maintaining their records for further processing sometimes it is hectic and buying specially developed software is not worth for small enterprises. The majority of the businesses has similar requirements and most of them use the traditional management system like recording data manually and maintaining hardcopies, a result, it consumes a lot of time as well as space. The proposed system is a web-based application specially built for small businesses like retailers and wholesalers. It is a standard business application made according to the standard requirements of businesses. The prime objective is to make data available for the user so that the user can access it anytime from anywhere and can modify it if need. Considering availability and accessibility this web-based application will help to achieve objectives. Our system focuses on the extraction of invoice data and segregates it into different parts like Vendor name, Invoice number, item name, and quantity. Our system tries to find out a better Management solution for auditing. We have initially focus on image invoices. The proposed solution follows three main steps preprocessing with the help of different OpenCV techniques and functions, then the next step is data extraction with the help of OCR technology and post-processing with the help of RegEx technology for better accuracy. The proposed system is web based applications, specially build for small businesses like retailers and wholesalers. It is a standard business application made according to the standard requirements of businesses. The prime objective is to make data available for the user so that the user can access it anytime from anywhere and can modify it if need. Considering availability and accessibility, this web-based application will help to achieve objectives.

2.8 Extraction of information from bill receipts using optical character recognition

Abstract:

This paper presents an application of optical character recognition (OCR) which can extract information from images of bills and receipts; store it as machine-processable text; in an organized manner for ease of access. It can do this efficiently even in the presence of watermarks on the bills or any shadows in the images of the bills. In developing this application, OpenCV has been used for the processing of the images and the Tesseract OCR engine has been used for optical character recognition and text extraction. The image is first processed using OpenCV for the removal of any shadows or watermarks present in it. For longer invoices, by employing the image bifurcation process, the data can be easily extracted which was not possible earlier. Furthermore, the processed image is passed on to the Tesseract OCR engine for the retrieving of text present in it. The text is then searched for important information, such as the total amount spent and the date on the receipt, using string processing.

2.9 Text extraction using OCR: A Systematic Review

Abstract:

In the digital era, almost everything is automated, and information is stored and communicated in digital forms. However, there are several situations where the data is not digitized, and it might become essential to extract text from those to store in digitized form. The latest technology such as Text recognition software has completely revolutionized the process of text extraction using Optical Character Recognition. Therefore, this paper introduces the concept, explains the process of extraction, presents the latest techniques, technologies, and current research in the area. Such a review will help other researchers in the field to get an overview of the technology.

2.10 Reagan: a low-level image processing network to restore compressed images to the original quality of JPEG

Abstract:

Low-level image processing is mainly concerned with extracting descriptions (that are usually represented as images themselves) from images. With the rapid development of neural networks, many deep learning-based low-level image processing tasks have shown outstanding performance. In this paper, we describe a unified deep learning based approach for low-level image processing, in particular, image denoising, image deblurring, and compressed image restoration. The proposed method is composed of deep convolutional neural and conditional generative adversarial networks. For the discriminator network, we present a new network architecture with bi-skip connections to address hard training and details losing issues. In the generative network, a multi-objective optimization is derived to solve the problem of common conditions being non-identical. Through extensive experiments on three low-level image processing tasks on both qualitative and

quantitative criteria, we demonstrate that our proposed method performs favorably against all current state-of-the-art approaches.

2.11 Offline optical character recognition (OCR) method: An effective method for scanned documents

Abstract:

Optical Character Recognition (OCR) is a major computer vision task by which characters of image are detected and recognized by comparing to training set images. Process of detecting character is one of the perplexing tasks in computer vision. This is because of input image often not correctly aligned or because of noise. This paper presents a complete Optical Character Recognition (OCR) system which is worked for English character mostly for Calibri font. This system first corrects skew of image if input image is not correctly aligned followed by noise reduction from input image. This process is passed through line and character segmentation that are passed into the recognition module and recognize characters. By experimenting with a set of 50 images, average achievement is 92%, 98% is for Calibri font. Moreover, the developed technique is computationally efficient and requires less time than other Optical character recognition system.

2.13 Deep Statistical Analysis of OCR Errors for Effective Post-OCR Processing

Abstract:

Post-OCR is an important processing step that follows optical character recognition (OCR) and is meant to improve the quality of OCR documents by detecting and correcting residual errors. This paper describes the results of a statistical analysis of OCR errors on four document collections. Five aspects related to general OCR errors are studied and compared with human-generated misspellings, including edit operations, length effects, erroneous character positions, real-word vs. non-word errors, and word boundaries. Based on the observations from the analysis we give several suggestions related to the design and implementation of effective OCR post-processing approaches.

2.14 Invoice Classification Using Deep Features and Machine Learning Techniques

Abstract:

Invoices are issued by companies, banks and different organizations in different forms including handwritten and machine-printed ones; sometimes, receipts are included as a separated form of invoices. In current practice, normally, classifying

these types is done manually, since each needs a special kind of processing such as making them suitable for optical character recognition systems (OCR). Classifying the invoices manually to different categories is a hard and time-consuming task. Therefore, we propose an automatic approach to classify invoices into three types: handwritten, machine-printed and receipts. The proposed method is based on extracting features using the deep convolutional neural network AlexNet. The features are classified using various machine learning algorithms, namely including Random Forests, K-nearest neighbors (KNN), and Naive Bayes. Different cross-validation approaches are applied in the experiments to ensure the effectiveness of the proposed solution. The best classification result was 98.4% (total accuracy), which was achieved by the KNN, such an almost perfect performance allows the proposed method to be used in practice as a preprocess for OCR systems, or as a standalone application.

2.15 Android-Based Text Recognition on Receipt Bill for Tax Sampling System

Abstract:

Text is an element which provides information to the readers. However, not all text is informative, some are still in need to be processed to generate information. In text processing process, data is required to be inputted into the system. The input process will be easier if the text is already in digital form. The main issue is when the text is in the non-digital form such as in the form of image. This image should be converted into a form which recognized by the machine. Therefore, an approach is required to be able to identify the text on the images, in expectation to generate text that can be processed by the machine. The method proposed for this research to identify the text is Convolutional Neural Network. Before and after entering the identification process, the input image will go through several pre-processing and post-processing phases to select which text to be displayed as a result. The testing process used images of receipt taken at the distance of 10cm and 12 cm. The result showed the accuracy rates of the testing using images of receipt taken at the distance of 10cm and 12 cm are 95% and 85% respectively.

2.16 A Proposed Approach for Character Recognition Using Document Analysis with OCR

Abstract:

Data entry has been a hectic job since the era of data accumulation started. Apparently, that is why there is a full-fledged career in data entry. Data entry method varies with requirements. Early days of computers relied on punch cards and

gradually keyboards and mouse came into picture which we still use. Touchscreens didn't take much time to replace the physical keyboard inputs and now as the result of human intelligence and innovation, the era of the optical input is here. Where the user does not even have to take the pain of thinking about entering data but can simply use an optical reader or scanner to input data. Using the computational power the individual elements like text, images, and special characters can be distinguished. OCR-Optical Character Recognizer does the work. OCR works similar to humans when it comes to character recognition as it maintains a database of characters and compares all the scanned elements with the database which makes it really simple to understand. This paper explains the working of an OCR in its different stages. That study helps in finding the various drawbacks of the conventional system. The paper also tells about how those shortcomings can be eliminated and how a better OCR that is future ready can be achieved.

2.17 Identification of Optimal Optical Character Recognition (OCR) Engine for Proposed System

Abstract:

A large number of research efforts have been put forward that attempts to transform a document image to format understandable for machine so that it can recognize the text or the information from the image. OCR i.e. Optical Character Recognition provides a solution for this. OCR is software that converts printed text and images into digitized form such that it can be manipulated by machine. OCR systems have established a niche place in pattern recognition. OCR has two categories, online and offline. The image of the scanned document goes through various stages like preprocessing, segmentation, feature extraction, etc. in order to retrieve the information from the image. OCR is also popular among the Android applications. Tesseract is one of the most widely used open-source library for implementing OCR in Android application.

2.18 Text Extraction from Bills and Invoices

Abstract:

This research tries to find out a methodology through which any data from the daily-use printed bills and invoices can be extracted. The data from these bills or invoices can be used extensively later on - such as machine learning or statistical analysis. This research focuses on extraction of final bill-amount, itinerary, date and similar data from bills and invoices as they encapsulate an ample amount of

information about the users purchases, likes or dislikes etc. Optical Character Recognition (OCR) technology is a system that provides a full alphanumeric recognition of printed or handwritten characters from images. Initially, OpenCV has been used to detect the bill or invoice from the image and filter out the unnecessary noise from the image. Then intermediate image is passed for further processing using Tesseract OCR engine, which is an optical character recognition engine. Tesseract intends to apply Text Segmentation in order to extract written text in various fonts and languages. Our methodology proves to be highly accurate while tested on a variety of input images of bills and invoices.

2.19 A different image content-based retrievals using OCR techniques

Abstract:

It is very difficult to retrieve image from large no of database which contain some message on it. The OCR techniques are becoming very efficient techniques for external and fast retrievals. The OCR technique search image based on data or text written on image. It search image or message contain in the image OCR the mechanical or electronic conversion of images of typed, handwritten or printed text, whether from scanned document, a scene-photo. For that Tesseract will use. After recognizing text from image by OCR, it will store that message in any file. After that we used Boyer-Moore string search algorithm, for searching string stored in file.

2.20 Inter-App Communication between Android Apps Developed in App-Inventor and Android Studio

Abstract:

Communications between mobile apps are an important aspect of mobile platforms. Android is specifically designed with inter-app communication in mind and depends on this to provide different platform specific functionalities. Android Apps can either be designed with the help of Android SDK and using IDEs such as Android Studio or by using a browser based platform called App Inventor. These two development platforms provide their own technique for inter-app communication in the same platform, however lack an established method of inter-app communication when apps are developed using the two separate development platforms. This paper provides the missing information required for the app communications and presents the method for sending and receiving arguments between apps developed in these two platforms. The paper also outlines the significance of the result, and examines their limitations.

2.21 An efficient mixed noise removal technique from grayscale images using noisy pixel modification technique

Abstract:

Removing or reducing noises from image is a very active research area in image processing domain. This paper presents an efficient noise removal technique to restore digital images corrupted by mixed noise, preserving image contents optimally. The proposed filtering technique consists of two steps: the noisy pixel detection step using fuzzy technique and the mixed noise filtering step. Noises addressed in this method are a combination of salt and pepper noise and Gaussian noise. This method reduces mixed noises considerably without compromising on edge sharpness. Experimental results show that the proposed technique consistently outperforms many existing fuzzy based algorithms while balancing the tradeoff between noise reduction and detail preservation. Hence, this mixed noise removal technique finds application in various segments of image processing like digital television, medical image processing, digital camera, surveillance systems etc.

2.24 Construction of Statistical SVM based Recognition Model for Handwritten Character Recognition**Abstract:**

There are many applications of the handwritten character recognition (HCR) approach still exist. Reading postal addresses in various states contains different languages in any union government like India. Bank check amounts and signature verification is one of the important application of HCR in the automatic banking system in all developed countries. The optical character recognition of the documents is comparing with handwriting documents by a human. This OCR is used for translation purposes of characters from various types of files such as image, word document files. The main aim of this research article is to provide the solution for various handwriting recognition approaches such as touch input from the mobile screen and picture file. The recognition approaches performing with various methods that we have chosen in artificial neural networks and statistical methods so on and to address nonlinearly divisible issues. This research article consisting of various approaches to compare and recognize the handwriting characters from the image documents. Besides, the research paper is comparing statistical approach support vector machine (SVM) classifiers network method with statistical, template matching, structural pattern recognition, and graphical methods. It has proved Statistical SVM

for OCR system performance that is providing a good result that is configured with machine learning approach. The recognition rate is higher than other methods mentioned in this research article. The proposed model has tested on a training section that contained various stylish letters and digits to learn with a higher accuracy level. We obtained test results of 91% of accuracy to recognize the characters from documents. Finally, we have discussed several future tasks of this research further.

2.25 Capsule Network Algorithm for Performance Optimization of Text Classification

Abstract:

In regions of visual inference, optimized performance is demonstrated by capsule networks on structured data. Classification of hierarchical multi-label text is performed with a simple capsule network algorithm in this paper. It is further compared to support vector machine (SVM), Long Short Term Memory (LSTM), artificial neural network (ANN), convolutional Neural Network (CNN) and other neural and non-neural network architectures to demonstrate its superior performance. The Blurb Genre Collection (BGC) and Web of Science (WOS) datasets are used for experimental purpose. The encoded latent data is combined with the algorithm while handling structurally diverse categories and rare events in hierarchical multi-label text applications.

2.26 Twitter Sentiment Analysis Using Supervised Machine Learning

Abstract:

Sentiment analysis aims to extract opinions, attitudes, as well as emotions from social media sites such as twitter. It has become a popular research area. The primary focus of the conventional way of sentiment analysis is on textual data. Twitter is the most renowned microblogging online networking site in which user posts updates related to different topics in the form of tweets. In this paper, a labeled dataset publicly available on Kaggle is used, and a comprehensive arrangement of pre-processing steps that make the tweets increasingly manageable to normal language handling strategies is structured. Since each example in the dataset is a pair of tweets and sentiment. So, supervised machine learning is used. In addition, sentiment analysis models based on naive Bayes, logistic regression, and support vector machine are proposed. The main intention is to break down sentiments all the more adequately. In twitter sentiment analysis, tweets are classified into positive sentiment and negative sentiment. This can be done using machine learning classifiers. Such classifiers will support a business, political parties, as well as analysts, etc., and so evaluate sentiments about them. By using training, data

machine learning techniques correctly classify the tweets. So, this method doesn't require a database of words, and in this manner, machine learning strategies are better and faster to perform sentiment analysis.

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

My project aim is to convert the invoice tabular contents into JSON and CSV formats with more than 95% accuracy. I achieved it through the basic image pre-processing techniques like black and white, grayscale and no-noise. After the conversion I read the characters from the invoice line by line using optical character recognition and then identify the company name and headers. Then clean the original text by removing all unnecessary characters. Based on the headers specific we are going to proceed with the logic and converted the clean text to JSON and CSV formats with 100% accuracy.

The prominent features of our protrude includes:

After acquiring the cleaned text from OCR, wordings are reshaped into the desired form, and then proceeded based on header-specific contents to convert it into JSON or CSV formats, respectively. The entire apparatus is set down as an app to make it

well ordered. The app will provide CSV and JSON configurations about invoices by giving the invoice number. The particulars may contain tabular contents and all the vital parts present in the invoice, and also end-user can give the input till where he needs the tabular contents [[Table. 1](#)].

CHAPTER 4

EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED

4.1 LANGUAGE USED:

- Python 3.9
- Java

4.2 PYTHON LIBRARIES USED:

- tesseract_OCR: this is used for specifying tesseract path (pip install tesseract).
- numpy: default library (pip install numpy).
- cv2: (pip install opencv-python).

- pdf2image. Converts pdf format to image if the invoice is in pdf.
- Fuzzywuzzy: it makes the image diaphanous.
- copy: to remove replicates(deepcopy).
- re: It is used for cleaning text.
- Json and csv.

The study started with the fascinating computer vision technology, initialized with OCR. The machine tries to understand objects present in the image with sublime parasite OCR. Several papers are referred to understand how OCR works mainly. After taking the ideology, the protrude is initialized using Tesseract OCR and got more than eighty percent accuracy. The activity starts by taking pdf or image as an input, further sent to a greater extent called image pre-processing. Will process the text additionally to remove obscure characters and split it by line to simplify it. Then will be converted into JSON and CSV. Finally, it can be viewed within the app.

Here are the in-detail recessions used:

A. Tesseract OCR

The most ranked and the best optical character recognition came into the public eye in the late 90s. The Tesseract OCR extracts the text out of the bounds from the respective image. This engine can be compatible with any source and is included in a python library as pytesseract.

B. Python libraries

Starting with NumPy, it is used for manipulating pixels into an array. It will be converted into collections for identification to an extent greater use.

cv2 is used for distinguishing the text present in the image by applying thresholding methods.

Poppler library is used in our setup to convert pdf as an image to enhance the standard. Link is provided to load in [\[Poppler\]](#).

Fuzzywuzzy makes the image diaphanous. The other significant use is to match the strings inside a list.

Regular expressions are used to match and remove unwanted characters.

Deepcopy is used to remove duplicates.

JSON: is used in the server to return objects as a key pair value.

CSV can be easily stored on the database and understood by everyone.

Pre-processed images

Grayscale image:

Grayscale is the primary image processing technique that makes the base to other

Description of Goods	HSN/SAC	Quantity	Rate	per	Amount
GARBAGE BAG (LARGE)		12 Nos	45.00	Nos	540.00
Life Boy Soap 10rs		12 Nos	8.47	Nos	101.64
S Hypo Chloried	28289019	2,000 KGS	30.00	KGS	60.00
Tide Powder 1kg@28%	3402	4 Nos	83.05	Nos	332.20
WHEEL POWDER 1KG		2 Nos	42.37	Nos	84.74
Mop Refill		1 Nos	120.00	Nos	120.00
Brooms		6 Nos	75.00	Nos	450.00
					1,688.58
Output CGST @ 9 %				9 %	111.48
Output SGST @ 9 %				9 %	111.48
Round Off					0.46
Total					₹ 1,912.00

Amount Chargeable (in words)
 INR One Thousand Nine Hundred Twelve Only

E. & O.E

Inverted image:

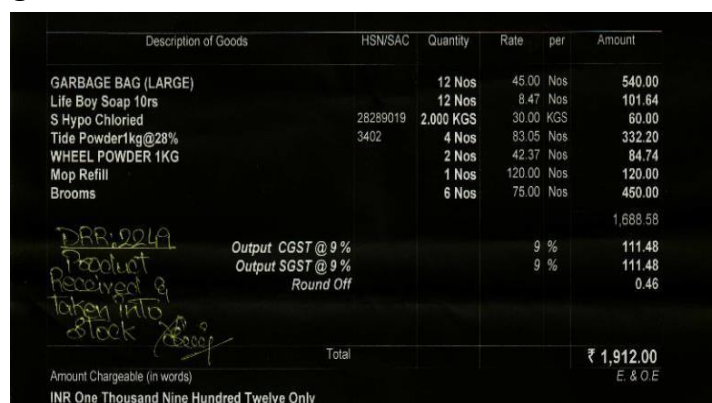


Fig 4.2: Inverted image

Canny image:

Canny is mainly used to crop out the unessential content in the image. When the image is unclear, this is a handy tool that comes in place and upgrades the picture, as depicted in fig. 4.3.

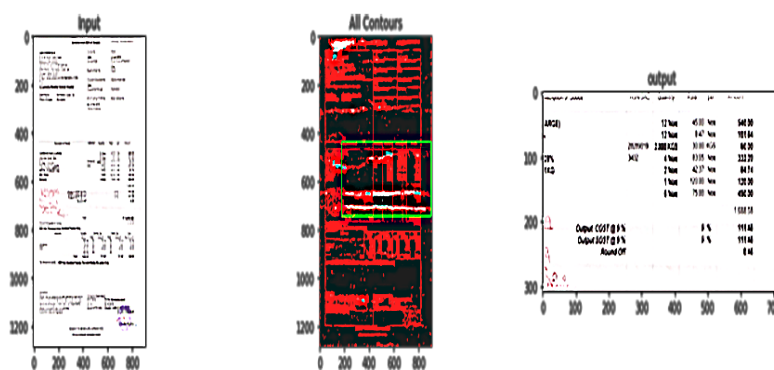


Fig 4.3: Canny image

Black and white image:

Black and white image initially takes Grayscale as an input to make the text bold and legible, as portrayed in fig. 4.4.

The entire pre-processing of the images is shown in flow chart.

Description of Goods	HSN/SAC	Quantity	Rate	per	Amount
GARBAGE BAG (LARGE)		12 Nos	45.00	Nos	540.00
Life Boy Soap 10rs		12 Nos	8.47	Nos	101.64
S Hypo Chloried	28289019	2.000 KGS	30.00	KGS	60.00
Tide Powder 1kg@28%	3402	4 Nos	83.05	Nos	332.20
WHEEL POWDER 1KG		2 Nos	42.37	Nos	84.74
Mop Refill		1 Nos	120.00	Nos	120.00
Brooms		6 Nos	75.00	Nos	450.00
					1,688.58
Output CGST @ 9 %			9 %		111.48
Output SGST @ 9 %			9 %		111.48
Round Off					0.46
		Total			₹ 1,912.00
Amount Chargeable (in words)					E & O.E
INR One Thousand Nine Hundred Twelve Only					

Fig 4.4: Black and white image

A. File picker app

The app which runs on the apache server takes the input from the user, as shown in sample Input can be in the form of a pdf or an image. XAMPP, which provides apache to run the app is used. A laptop is turned into a XAMPP server, which only works on a local area network. After taking the input, it will be uploaded to the database and the local storage. The local storage gets the information with the help of retrofit, which connects us with an HTTP request.

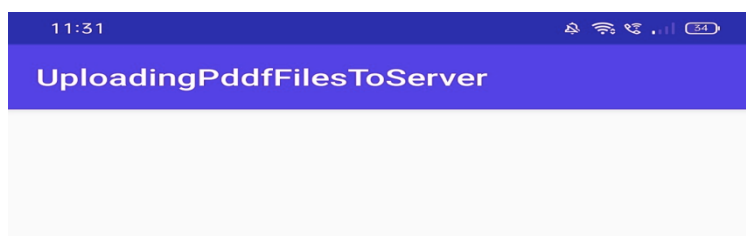


Fig 4.5: App interface

B. Database

The database used is MySQL. Eventually, the output will get stored inside MySQL on which the server is running. The distinguished database used is cost- effective and can keep it in local storage, making the python script much more accessible.

Name	Date modified	Type	Size
documents	11-12-2021 08:13 PM	File folder	
images	12-12-2021 07:12 PM	File folder	
db_config.php	11-12-2021 11:33 PM	PHP Source File	1 KB
download_image.php	07-07-2020 11:13 AM	PHP Source File	1 KB
downloadDocument.php	07-07-2020 11:13 AM	PHP Source File	1 KB
getStudent.php	07-07-2020 11:13 AM	PHP Source File	1 KB
hi.php	12-12-2021 01:43 PM	PHP Source File	2 KB
Jain_Chemicals.json	12-12-2021 06:49 PM	JSON Source File	3 KB
Names.csv	12-12-2021 06:49 PM	Microsoft Excel C...	1 KB
save_user.php	07-07-2020 11:13 AM	PHP Source File	1 KB
Unitron.json	12-12-2021 06:59 PM	JSON Source File	2 KB
upload_document.php	12-12-2021 01:37 PM	PHP Source File	1 KB
upload_image.php	07-07-2020 11:13 AM	PHP Source File	1 KB

Fig 4.6: System local storage

C. PHP

The personal home page (PHP) script is a piece of cake to make our back-end python code run with the help of a command shell to store the output for further processing.

D. Python script, which runs on the back-end

Code is automated with PHP's help, which runs along with the app whenever the input is obtained from the user. The command shell is opened up with the PHP script's collateral running and tells us the exact time on how long the program ran. The whole scenario depicts the complexity of the project in words. The forthcoming section describes the phases implemented with the accrued libraries written above to amalgamate the consequence.

4.3 Phases of work

PHASE 1. App enacts as a front-end

With the help of android tools, the fundamental file picker app takes the input in a pdf or an image file. An app that runs on the XAMPP server takes the input and stores it inside the database and the predominant storage on our laptop. Now, send the file to the python script to lay a path for the coming procedures.

PHASE 2. PHP connected with a python script which runs as a back-end

Whenever the invoice is given to the app, it will be mechanized with the python script. Along with the assistance of PHP, the command shell is opened simultaneously and stores the file to automate the code.

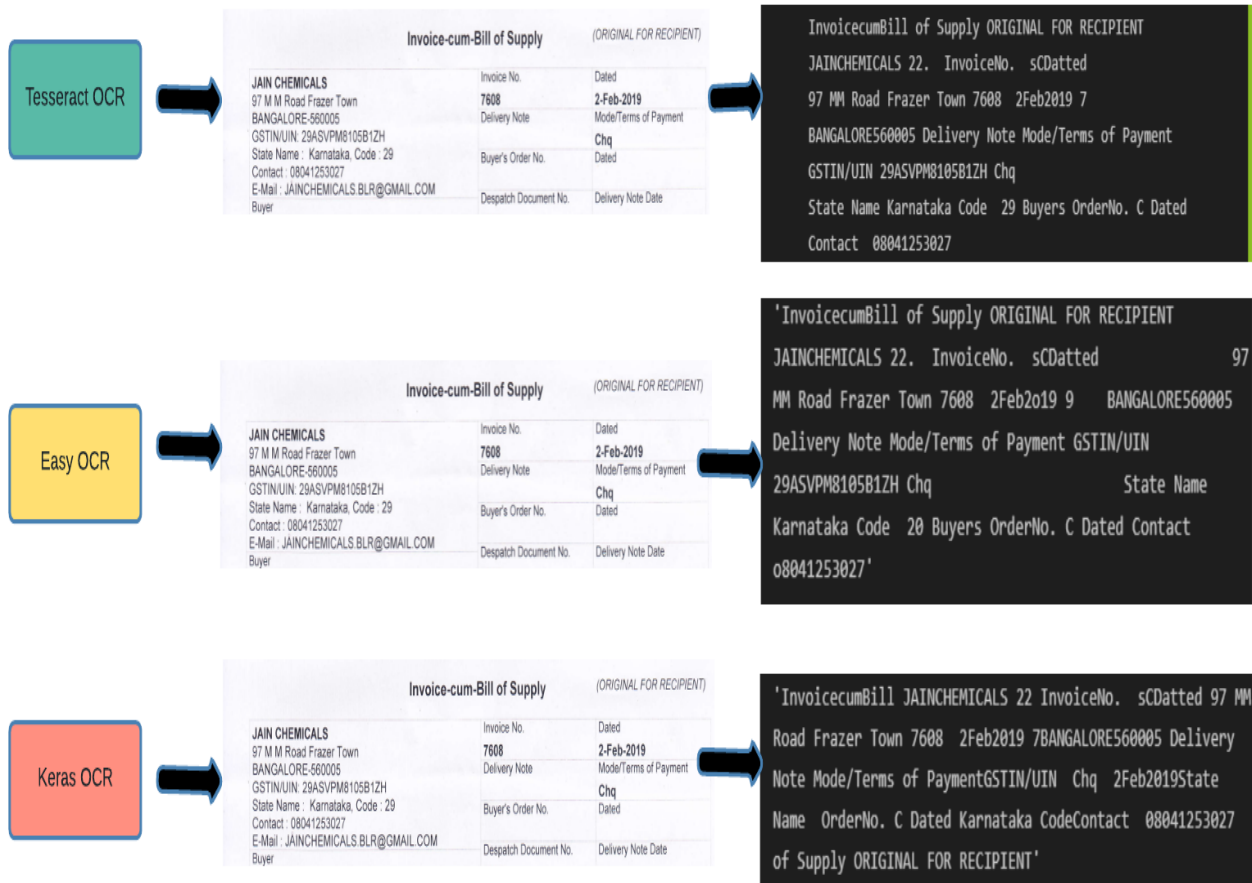


Fig 4.7: OCR results

PHASE 3. Receiving the input and reading the text

If the input is in the form of a pdf, if it is an image, it will directly get into the pre-processing phase. The initial step after getting the image is to digitize the image to make the machine understand the text and compare it with the empirical database to get a constitutional output. The program which will convert the image into the necessary format is OpenCV.

OpenCV provides many resources before going into image processing. The most crucial method is thresholding, where the image is contoured and converted into

binary format. The next step is to imply various image pre-processing for the image got. The image needs to be converted into Grayscale to pave the way for the remaining techniques. Essentially the image will go through inverted processing as the accuracy is not good. The following methodology is noise removal, as it discards the noise around and makes the picture look good.

Right after that, thick font and thin font are applied, which are not effectual, which leads to abate in the accuracy. In the coming step, canny, which is good at edge detection and contours is used. It tries to fan out unessential parts of the picture and pass it on to the black and white to make by tesseract work on it.

After the image is converted into OpenCV format, image cleaning is employed, which tries to improve the grade of the image for better results.

Finally, the image is passed on to the OCR to detect text.

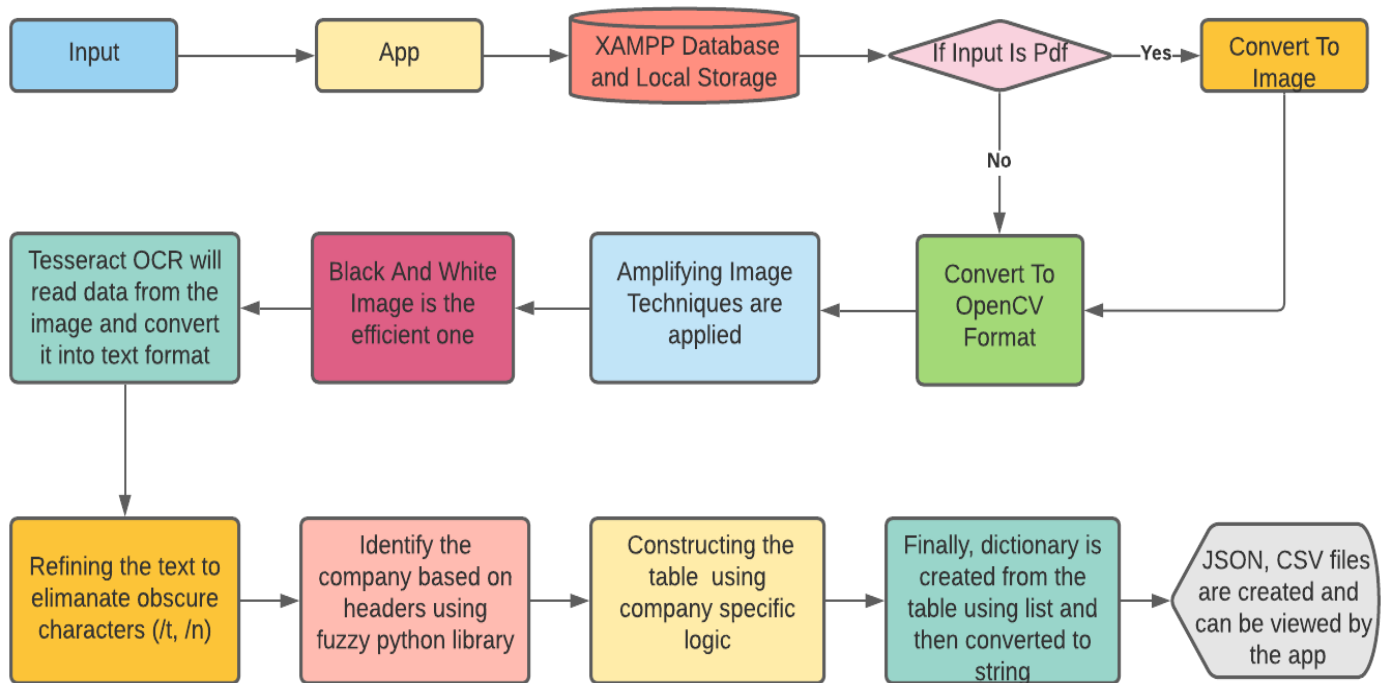


Fig 4.8: Flow chart

PHASE 4. Tesseract OCR

As depicted in fig.7, out of the three OCRs, the Tesseract OCR is used, which is the finest of all the remaining OCRs as it goes based on invoice format. Once merged with python libraries, it is called the Pytesseract OCR and can be easily turned with any non-proprietary library.

Tesseract OCR mainly works on recurrent neural network (RNN) called Forward Long short-term memory which is a neural network subsystem customized as a line recognizer of text as it is the best method. In a pictorial representation of how the tesseract OCR uses long short-term memory network which is initiated in a forward manner.

Firstly, it will identify each and every letter and compare it with the beam search which goes sequentially one after the another. After the extraction, all the unwanted symbols like \t will be coming up. In the further section, cleaning the text has been discussed.

PHASE 5. Text cleaning and Table Detection

After getting the OCR text, this phase removes obscure characters with the help of regular expressions. In the tertiary step, the text-based online segmentation is split.

As the cleaned text is in a string form, the text must be transformed into a list of items. Succeeding the previous step, the header should be popped out based on the fuzzy library, which helps to match the manual header. If the contents match more than eighty percent, a ratio is set to take out the elements.

Preceding the above point, the text is segregated based on headers, the contents to pop out are identified and a new list is created.

If the contents of the cleaned text match, this will go into the distributor-specific logic to extract the table. After identifying header contents as shown in list created picture, it will search for the keywords specified and place the table's onset and outcome. Another method is to determine based on serial number (S.I. No.). For instance, it will take the final serial number of the tabular column and consider it as the final numeric of the table leading to end of the table.

Suppose if the invoice does not contain the S.I. No, it will identify the keyword given in the code and find the end of the table accordingly.

PHASE 6. Recreation of Table and Storing into JSON and CSV format

As portrayed in list created picture, the cleaned and extracted text will often be different, and missing fields will be ignored. The underlined text in list created picture is converted into the preferred configuration. This phase will match the contents into specific columns based on company-specific logic.

Once the tables have been recreated, the contents will be saved as JSON and CSV files, respectively. These files are formed with the help of JSON dump python library

by creating dictionary and zipping it further into a CSV with the assistance of JSON dump along with CSV dictionary writer which is a module in python.

In order to create a dictionary, a specific logic to segregate the columns by creating a range of keywords is written. This sorting works on similar formats if there are no misplaced values. Handwritten characters can also be identified based on the legibility.

PHASE 7. To view the contents using the app

The output will be stored inside the local storage where the PHP script is placed. After receiving the outcome, it is uploaded to the XAMPP database through which our operation is connected and formats can be viewed easily with an app. The App should be discovered by the system to view the formats saved.

A. Algorithm

Step 1: Firstly, the invoice is taken from the end-user through an app that runs on the XAMPP server. The source can be in pdf or an image format. Then, it is converted into a processed image for more accuracy and stored in the database.

Step 2: In the second step, image processing techniques are applied and the best one which suits is found. As the python script runs on the back-end, with the help of PHP script, code can be connected to the java app.

Step 3: In the tertiary step, Tesseract OCR will come in place to extract the text from the invoice and remove flawed characters to make it into a list.

Step 4: To identify the distributor, the header contents are used to segregate with one another.

Step 5: Then, the end of the table is found with keywords and S.I. No.

Step 6: Finally, the table is extracted and converted into a JSON and CSV file.

Step 7: Contents can be finally viewed through the app.

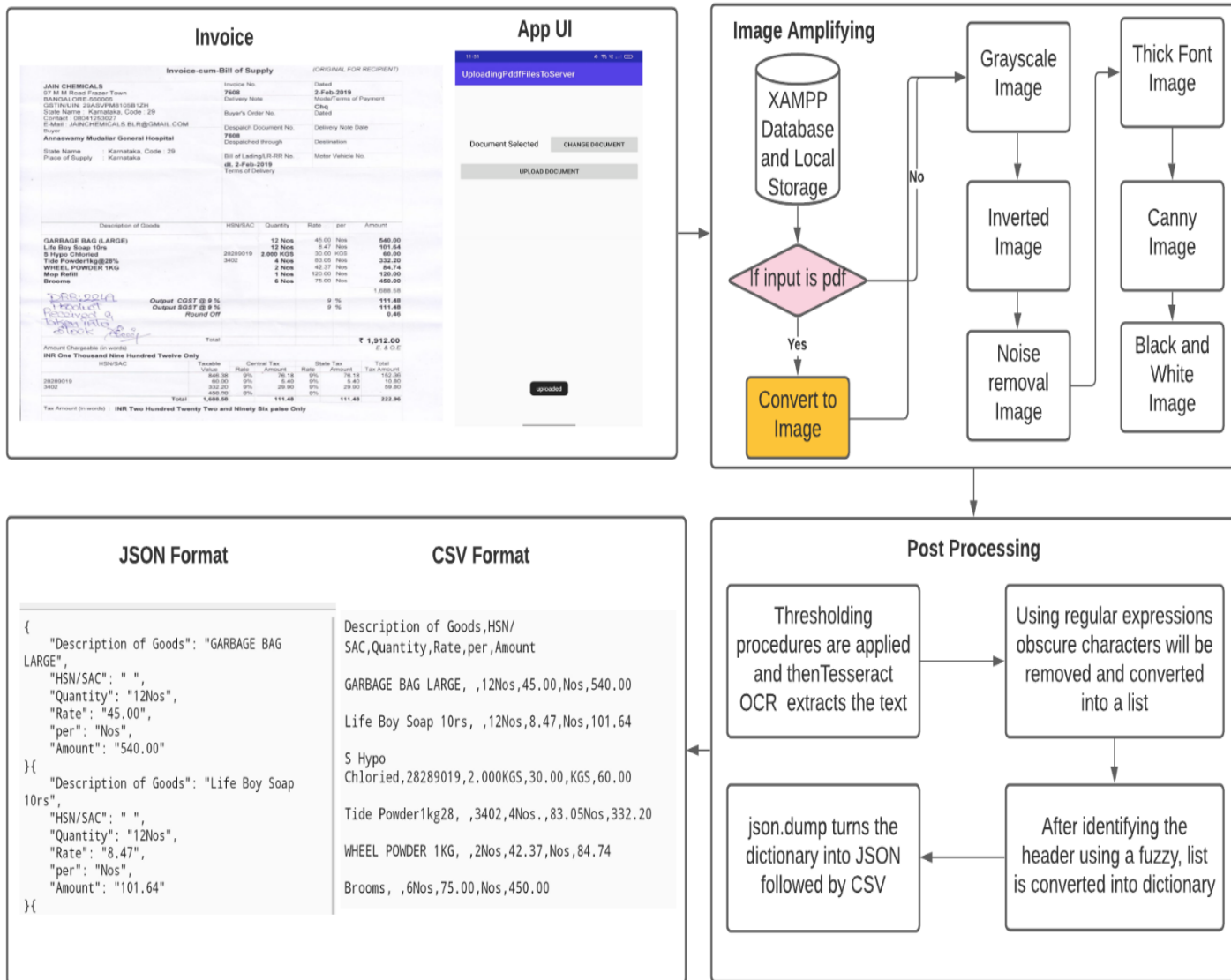


Fig 4.8: System architecture

CHAPTER 5

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

This segment summarizes the outcome of our protrude. Primarily, the results are compared with the existing OCRs end product and bottom line to enhance the text's conversion into JSON and CSV formats.

5.1 Experimental results compared with in-use OCR results

The existing OCRs have the outcome in the form of cleaned text and some other focus on qualitative analysis with great accuracy. Major OCRs try to extract essential fields present in the invoice as a potentially significant event, as shown in Considering the gravity of the situation, an adamant result by turning text into a desired form is given. With a ninety percent accuracy, the result is articulated.

5.2 Subjective results

The protrude is aggrandized with a qualitative idea of converting the text into JSON and CSV formats, as portrayed in JSON and CSV formats. The JSON file can return the particulars to the web page. As in JSON format key pair values are generated. The other main thing is the comma-separated value (CSV) which shows the contents of the invoice in a table as portrayed in CSV format. With the above result, this model can be used in several ways for invoice data extraction. Table 1 explains about the accuracy of the sample input. The accuracy precision is calculated based on matching strings by taking sample text of each element in the table separately and the OCR extracted text. Areas to address are: missing values, misprinted values and misplaced values. Character accuracy is evaluated by the number of actual characters with their positions which will split up by the aggregate of actual characters to give the percentage value.

Sl.No	Accuracy results for sample input		
	<i>Input value</i>	<i>OCR value</i>	<i>Accuracy percentage</i>
1.	Headers 'Description of Goods HSN/SAC Quantity Rate per Amount'	Headers 'Description of Goods HSN/SAC Quantity Rate per Amount'	100%
2	Tabular contents 'GARBAGE BAG (LARGE) 12 Nos 45.00 Nos 540.00'	Tabular contents 'GARBAGE BAG LARGE 12 Nos 45.00 Nos 540.00'	98%
3	'Life Boy Soap 10rs 12 Nos 8.47 Nos 101.64'	'Life Boy Soap 10rs 12 Nos 8.47 Nos 101.64'	100%
4	'S Hypo Chloried 28289019 2.000 KGS 30.00 KGS 60.00'	'S Hypo Chloried 28289019 2.000 KGS 30.00 KGS 60.00'	100%
5	'Tide Powder1kg28 3402 4Nos. 83.05 Nos 332.20'	'Tide Powder1kg28 3402 4Nos. 83.050 Nos 332.20'	98%
6	'WHEEL POWDER 1KG 2 Nos 42.37 Nos 84.74'	'WHEELPOWDER 1KG 2 Nos 42.37 Nos 84.74'	98%
7	'Brooms 6 Nos 75.00 Nos 450.00 '	'Brooms 6 Nos 75.00 Nos 450.00 '	100%
Total Accuracy			99%

Table: 1 Accuracy analysis

5.3 LIMITATIONS:

- If any Handwritten text in invoice this will affect the dialectics of the program.
- When an image is Fuzzy.
- If OCR detects wrong text, then these dialectics will not work.

- If headroom is much bigger between the end of the table and keyword that we are searching for in the invoice.
- Finally, if there are any misprinted values in the invoice, then the JSON will be affected.
- Only erect pdf will work.

CHAPTER 6

SUMMARY AND CONCLUSIONS

In this fast-paced world, to match the needs of grieving people, An OCR is put forward in this paper to extract the text inside the image. The affiliate Computer vision technology tries to lend a helping hand which initializes this protrude. It uses various image processing techniques like converting the given image into Grayscale and then sending it to threshold the pixels. The tests are generated based on a determined number of invoices. As far as the OCR is bothered, Tesseract is considered, which gave us an appropriate result. Relinquish is observed initially, but black and white gave us the conclusion with great accuracy. After that, the text is cleaned with regular expressions to pass the text through phases. In the sequential step, the reader gets into creating JSON and CSV configurations. The main limitation is it only works on the format specified in the program and only restricted to English language. To enhance the project further, an OCR needed to be structured to identify spaces, and the text can be segregated based on distances. The next possible solution is to enlarge and classify performance on several invoices. The main basis can be taken form this paper to implement it for other languages but mostly bills will be in the universal language itself. This experimental setup can be helpful for formats similar to the sample invoice, as shown in sample invoice. With the help of an app, the end-users can integrate with this module and ease their work.

REFERENCES

- [1] Jiju, Alan, Shaun Tuscano, and Chetana Badgujar. "OCR text extraction." *International Journal of Engineering and Management Research* 11.2 (2021): 83-86.
- [2] Jung, Seokweon, et al. "Mixed-Initiative Approach to Extract Data from Pictures of Medical Invoice." *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*. IEEE, 2021.
- [3] R. Sharma, P. Dave, and J. Chaudhary, "OCR for Data Retrieval: An analysis and Machine Learning Application model for NGO social volunteering," 2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics, and Cloud) (I-SMAC), 2021, pp. 422-427, DOI: 10.1109/I-SMAC52330.2021.9640890.
- [4] A. Revathi and N. A. Modi, "Comparative Analysis of Text Extraction from Color Images using Tesseract and OpenCV," 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom), 2021, pp. 931- 936, DOI: 10.1109/INDIACom51348.2021.00167.
- [5] K. M. Yindumathi, S. S. Chaudhari and R. Aparna, "Analysis of Image Classification for Text Extraction from Bills and Invoices," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1-6, DOI: 10.1109/ICCCNT49239.2020.9225564.
- [6] Z. Zhou and L. Lin, "Text Orientation Detection Based on Multi Neural Network," 2020 Chinese Automation Congress (CAC), 2020, pp. 6175-6179, DOI: 10.1109/CAC51589.2020.9327425.
- [7] M. S. Satav, T. Varade, D. Kothavale, S. Thombare, and P. Lokhande, "Data Extraction from Invoices Using Computer Vision," 2020 IEEE 15th International Conference on Industrial and Information Systems (ICES), 2020, pp. 316-320, DOI:

10.1109/ICIIS51140.2020.9342722.

[8] V. Kumar, P. Kaware, P. Singh, R. Sonkusare and S. Kumar, "Extraction of information from bill receipts using optical character recognition," 2020 International Conference on Smart Electronics and Communication (ICOSEC), 2020, pp. 72-77, DOI: 10.1109/ICOSEC49089.2020.9215246.

[9] R. Mittal and A. Garg, "Text extraction using OCR: A Systematic Review," 2020 Second International Conference on Inventive Research in Computing Applications (CIRCA), 2020, pp. 357-362, DOI: 10.1109/ICIRCA48905.2020.9183326.

[10] Zhu C, Chen Y, Zhang Y, Liu S, Li G (2019) Reagan: a low-level image processing network to restore compressed images to the original quality of JPEG. In: 2019 Data Compression Conference (DCC). IEEE, pp 616.

[11] M. Rahman Majumder, B. Uddin Mahmud, B. Jahan, and M. Alam, "Offline optical character recognition (OCR) method: An effective method for scanned documents," 2019 22nd International Conference on Computer and Information Technology (ICCIT), 2019, pp. 1-5, DOI: 10.1109/ICCIT48885.2019.9038593.

[12] _Android_Studio, <http://developer.android.com/tools/studio/index.html>.

[13] T. -T. -H. Nguyen, A. Jatowt, M. Coustaty, N. -V. Nguyen and A. Doucet, "Deep Statistical Analysis of OCR Errors for Effective Post-OCR Processing," 2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL), 2019, pp. 29-38, DOI: 10.1109/JCDL.2019.00015.

[14] A. S. Tarawneh, A. B. Hassanat, D. Chetverikov, I. Lendak, and C. Verma, "Invoice Classification Using Deep Features and Machine Learning Techniques," 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), 2019, pp. 855-859, DOI: 10.1109/JEEIT.2019.8717504.

[15] R. F. Rahmat, D. Gunawan, S. Faza, N. Haloho, and E. B. Nababan, "Android-Based Text Recognition on Receipt Bill for Tax Sampling System," 2018 Third International Conference on Informatics and Computing (ICIC), Palembang, Indonesia, 2018, pp. 1-5, DOI: 10.1109/IAC.2018.8780416.

[16] H. Singh and A. Sachan, "A Proposed Approach for Character Recognition Using Document Analysis with OCR," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018, pp. 190-195, DOI: 10.1109/ICCONS.2018.8663011.

[17] M. G. Marne, P. R. Futane, S. B. Kolekar, A. D. Lakhadive, and S. K. Marathe, "Identification of Optimal Optical Character Recognition (OCR) Engine for Proposed System," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, 8585.

[18] H. Sidhwa, S. Kulshrestha, S. Malhotra and S. Virmani, "Text Extraction from Bills and Invoices," 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2018, pp. 564-568, DOI: 10.1109/ICACCCN.2018.8748309.

[19] P. A. Wankhede and S. W. Mohod, "A different image content-based retrievals using OCR techniques," 2017 International conference of Electronics, Communication, and Aerospace Technology (ICECA), 2017, pp. 155-161, DOI: 10.1109/ICECA.2017.8212785.

- [20] L. Allison and M. M. Fuad, "Inter-App Communication between Android Apps Developed in App-Inventor and Android Studio," 2016 IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft), 2016, pp. 17-18, DOI: 10.1109/MobileSoft.2016.018.
- [21] Jayasree M. and N. K. Narayanan, "An efficient mixed noise removal technique from grayscale images using noisy pixel modification technique," 2015 International Conference on Communications and Signal Processing (ICCSP), 2015, pp. 0336-0339, DOI: 10.1109/ICCSP.2015.7322901.
- [22] An Overview of the Tesseract OCR Engine - Research at Google. <https://research.google.com/pubs/archive/33418.pdf>.
- [23] OpenCV available at <https://opencv.org/> accessed on Nov 2017.
- [24] Hamdan, Yasir Babiker. "Construction of Statistical SVM based Recognition Model for Handwritten Character Recognition." Journal of Information Technology 3, no. 02 (2021): 92-107.
- [25] Manoharan, J. Samuel. "Capsule Network Algorithm for Performance Optimization of Text Classification." Journal of Soft Computing Paradigm (JSCP) 3, no. 01 (2021): 1-9.
- Yadav, Nikhil, Omkar Kudale, Aditi Rao, Srishti Gupta, and Ajitkumar Shitole. "Twitter Sentiment Analysis Using Supervised Machine Learning." In Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2020, pp. 631-642. Springer Singapore, 2021.

APPENDIX:

A: Screenshots:-

Sample Invoice

(ORIGINAL FOR RECIPIENT)

JAIN CHEMICALS 97 M M Road Frazer Town BANGALORE-560005 GSTIN/UIN: 29ASVPM8105B1ZH State Name : Karnataka, Code : 29 Contact : 08041253027 E-Mail : JAINCHEMICALS.BLR@GMAIL.COM Buyer Annaswamy Mudaliar General Hospital State Name : Karnataka, Code : 29 Place of Supply : Karnataka		Invoice No. 7608 Delivery Note Buyer's Order No. Despatch Document No. 7608 Despatched through Bill of Lading/LR-RR No. dt. 2-Feb-2019 Terms of Delivery		Dated 2-Feb-2019 Mode/Terms of Payment Chq Dated Delivery Note Date Destination Motor Vehicle No.	
---	--	--	--	--	--

Description of Goods	HSN/SAC	Quantity	Rate	per	Amount
GARBAGE BAG (LARGE)		12 Nos	45.00	Nos	540.00
Life Boy Soap 10rs		12 Nos	8.47	Nos	101.64
S Hypo Chlorid	28289019	2.000 KGS	30.00	KGS	60.00
Tide Powder 1kg@28%	3402	4 Nos	83.05	Nos	332.20
WHEEL POWDER 1KG		2 Nos	42.37	Nos	84.74
Mop Refill		1 Nos	120.00	Nos	120.00
Brooms		6 Nos	75.00	Nos	450.00
					1,688.58
				9 %	111.48
				9 %	111.48
					0.46
Total					₹ 1,912.00

Amount Chargeable (in words) E. & O.E

INR One Thousand Nine Hundred Twelve Only

HSN/SAC	Taxable Value	Central Tax		State Tax		Total Tax Amount
		Rate	Amount	Rate	Amount	
	846.38	9%	76.18	9%	76.18	152.36
28289019	60.00	9%	5.40	9%	5.40	10.80
3402	332.20	9%	29.90	9%	29.90	59.80
	450.00	0%		0%		
Total	1,688.58		111.48		111.48	222.96

Tax Amount (in words) : **INR Two Hundred Twenty Two and Ninety Six paise Only**

42

txt

✓ 0.4s

"Invoice-cum-Bill of Supply (ORIGINAL FOR RECIPIENT)\nJAIN CHEMICALS i Invoice No.
Dated\n97 MM Road Frazer Town 7608 2-Feb-2019\nBANGALORE-560005 Delivery Note Mode/Terms
of Payment\n' GSTIN/UIN: 29ASVPM8105B1ZH Chq\nState Name: Karnataka, Code : 29 Buyer's
Order No. - Dated\nContact : 08041253027\nnaha eee Despatch Document No. Delivery Note
Date -\n: . 7608\nAnnaswamy Mudaliar General Hospital Despatched through
Destination"\nState Name : Karnataka, Code : 29 _ ee\nPlace of Supply : Karnataka f Bill
of Lading/LR-RR No. Motor Vehicle No.\ndt. 2-Feb-2019\nTerms of Delivery\nDescription of
Goods HSN/SAC = Quantity Rate per Amount\nGARBAGE BAG (LARGE) 12 Nos 45.00 Nos
540.00\nLife Boy Soap 10rs 12 Nos 8.47 Nos 101.64\nS Hypo Chloried 28289019 2.000 KGS
30.00 KGS 60.00\nTide Powdertkg@28% 3402 4 Nos 83.05 Nos 332.20\nWHEEL POWDER 1KG 2 Nos
42.37 Nos 84.74\nMop Refill 1 Nos 120.00 Nos 120.00\nBrooms 6 Nos 75.00 Nos 450.00 -
\n1,688.58\n| RR Uy Output CGST@9% 9% 111.48\n| Vmatuct Output SGST @9 % 9 %
111.48\nReccly ect @ Round Off 0.46\n'eke Tale.\nCMe HALO\nSS ae\noe cl Total a = 1
5912.00\nAmount Chargeable (in words) E.& OE\nINR One Thousand Nine Hundred Twelve
Only\n' ~ HSN/SAC 'Taxable —sCentralTax ss State Tax Total\nValue Rate Amount Rate
Amount Tax Amount\n846.38 9% 76.18 9% 76.18 152:36\n28289019 60.00 9% 5.40 9% 5.40
10.80\n3402 332.20 9% 29.90 9% 29.90 59.80\n450.00 0% 0% : \nae ee _Total 1,688.58 = | -
111.48 111.48 222.96\nTax Amount (in words) : INR Two Hundred Twenty Two and Ninety Six
paise Only\nDeclaration ;\nPlease use our products only after laboratory tests and is
Company's Bank Details\nSTRICTLY NOT FOR MEDICAL USE. Goods once sold Bank Name - KOTAK
MAHINDRA BANK\ncannot be exchanged or taken back. Our responsibility Ale No. :
1611875278\nceases once goods leave our premises. Branch & IFS Code : FRAZER TOWN & GEAR
BOB\nCustomer's Seal and Signature f eee CHEWHEALS\nVes\né 1) 0j.\n' _ oe - a ee
atorised Signatory _\nSUBJECT TO BENGALURU JURISDICTION *\nThis is a Computer Generated
Invoice .\n\x0c"

Cleaned text:

```
Interactive-1 X [Icons] ...
Clear All Restart Interrupt Variables Save Export Expand Collapse Python 3.9.2 64-bit

... Image conversion successful
ocrd successfully

['InvoicecumBill of Supply ORIGINAL FOR RECIPIENT', 'JAINCHEMICALS 22 InvoiceNo sCDatted ', '97 MM Road
Frazer Town 7608 2Feb2019 7', 'BANGALORE560005 Delivery Note Mode/Terms of Payment', 'GSTIN/UIN
29ASVPM8105B1ZH Chq', 'State Name Karnataka Code 29 Buyers OrderNo C Dated ', 'Contact 08041253027', 'BU
AINCHEMICATS BERGMAICOM Despatch Document No Delivery Note Date ', ' 7608', 'Annaswamy Mudaliar General
Hospital Deel Destination ', 'State Name Karnataka Code 29 a', 'Place of Supply Karnataka Bill of
Lading/LRRR No Motor Vehicle No', 'dt 2Feb2019', 'Terms of Delivery', '', 'Description of Goods HSN/SAC
Quantity Rate per Amount', 'GARBAGE BAG LARGE 12 Nos 4500 Nos 54000', 'Life Boy Soap 10rs 12 Nos 847 Nos
10164', 'S Hypo Chloried 28289019 2000 KGS 3000 KGS 6000', 'Tide Powder1kg28 3402 4Nos 8305 Nos 33220',
'WHEEL POWDER 1KG 2 Nos 4237 Nos 8474', 'Brooms 6 Nos 7500 Nos 45000 ', ' 168858', 'DRR Output CGST9 9
11148', 'VwDaluct Output SGST 9 9 11148', 'Kecoty eck G Round Off 046', 'Token fate', 'CAYO HALO ',
'Sleck os p ', 'me ssi AC 191200', 'Amount Chargeable in words EOE', 'INR One Thousand Nine Hundred
Twelve Only', 'a HSN/SAC Taxable ss CentralTax ss State Tax sSCSsTTtl', 'Value Rate Amount Rate Amount
Tax Amount', '84638 9 7618 9 7618 15236', '28289019 6000 9 540 9 540 1080', '3402 33220 9 2990 9 2990 5980
', '45000 0 0', 'HO ee Total 168858 11148 11148 22296', 'Tax Amount in words INR Two Hundred Twenty
Two and Ninety Six paise Only', 'Declaration ', 'Please use our products only after laboratory tests and
is Companys Bank Details ', ' STRICTLY NOT FOR MEDICAL USE Goods once sold Bank Name KOTAK MAHINDRA
BANK', 'cannot be exchanged or taken back Our responsibility Alc No 1611875278 ', 'ceases once goods
leave our premises Branch IFS Code FRAZER TOWN KKSkg80BOBI', 'Customers Seal and Signature TAN
CHEWHEALS', 'a VR', ' CB GBROISEM BignAtory C', 'SUBJECT TO BENGALURU JURISDICTION ', 'This is a Computer
Generated Invoice', '']

Table extracted

--- 9.01702356338501 seconds ---
```

JSON format:

```
main > {} Jain Chemicals.json > ...
1  {}
2  "Description of Goods": "GARBAGE BAG LARGE",
3  "HSN/SAC": " ",
4  "Quantity": "12Nos",
5  "Rate": "45.00",
6  "per": "Nos",
7  "Amount": "540.00"
8  {}{
9  "Description of Goods": "Life Boy Soap 10rs",
10 "HSN/SAC": " ",
11 "Quantity": "12Nos",
12 "Rate": "8.47",
13 "per": "Nos",
14 "Amount": "101.64"
15 }{
16 "Description of Goods": "S Hypo Chloried",
17 "HSN/SAC": "28289019",
18 "Quantity": "2.000KGS",
19 "Rate": "30.00",
20 "per": "KGS",
21 "Amount": "60.00"
22 }{
23 "Description of Goods": "Tide Powdertkg28",
24 "HSN/SAC": "3402",
25 "Quantity": "4Nos",
26 "Rate": "83.05",
27 "per": "Nos",
28 "Amount": "332.20"
29 }{
30 "Description of Goods": "WHEEL POWDER 1KG",
31 "HSN/SAC": " ",
32 "Quantity": "2Nos",
33 "Rate": "42.37",
34 "per": "Nos",
35 "Amount": "84.74"
36 }{
37 "Description of Goods": "Mop Refill",
38 "HSN/SAC": " "
```

CSV Format:

	A	B	C	D	E	F	
1	Description of Goods	HSN/SAC	Quantity	Rate	per	Amount	
2	GARBAGE BAG LARGE		12Nos	45	Nos	540	
3	Life Boy Soap 10rs		12Nos	8.47	Nos	101.64	
4	S Hypo Chloried	28289019	2.000KGS	30	KGS	60	
5	Tide Powdertkg28	3402	4Nos	83.05	Nos	332.2	
6	WHEEL POWDER 1KG		2Nos	42.37	Nos	84.74	
7	Mop Refill		1Nos	120	Nos	120	
8	Brooms		6Nos	75	Nos	450	
9							

B: Source code:

ANDROID APP CODE:

MainActivity.java

```
package com.example.uploadingpddffilestoserver;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Base64;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class MainActivity extends AppCompatActivity {

    private Button btnSelect, btnUpload;
    private TextView textView;

    private int REQ_PDF = 21;
    private String encodedPDF;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textView = findViewById(R.id.textView);
        btnSelect = findViewById(R.id.btnSelect);
        btnUpload = findViewById(R.id.btnUpload);

        btnSelect.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent chooseFile = new Intent(Intent.ACTION_GET_CONTENT);
                chooseFile.setType("application/pdf");

                chooseFile = Intent.createChooser(chooseFile, "Choose a
file");

                startActivityForResult(chooseFile, REQ_PDF);
            }
        });
    }
}
```

```

    }
    });

    btnUpload.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            uploadDocument();
        }
    });

}

private void uploadDocument() {

    Call<ResponsePOJO> call =
RetrofitClient.getInstance().getAPI().uploadDocument(encodedPDF);
    call.enqueue(new Callback<ResponsePOJO>() {
        @Override
        public void onResponse(Call<ResponsePOJO> call,
Response<ResponsePOJO> response) {
            Toast.makeText(MainActivity.this,
response.body().getRemarks(), Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onFailure(Call<ResponsePOJO> call, Throwable t) {
            Toast.makeText(MainActivity.this, "file uploaded",
Toast.LENGTH_SHORT).show();
        }
    });
}

@Override
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if(requestCode == REQ_PDF && resultCode == RESULT_OK && data !=
null){

        Uri path = data.getData();

        try {
            InputStream inputStream =
MainActivity.this.getContentResolver().openInputStream(path);
            byte[] pdfInBytes = new byte[inputStream.available()];
            inputStream.read(pdfInBytes);
            encodedPDF = Base64.encodeToString(pdfInBytes,
Base64.DEFAULT);

            textView.setText("Document Selected");
            btnSelect.setText("Change Document");

            Toast.makeText(this, "Document Selected",
Toast.LENGTH_SHORT).show();

        } catch (IOException e) {

```



```

        e.printStackTrace();
    }

    }

}

```

MainActivity.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="111dp"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="9dp"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/textView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:gravity="center"
                android:text="Not Selected"
                android:textColor="@android:color/background_dark"
                android:textSize="18dp" />

            <Button
                android:id="@+id/btnSelect"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="Select Document" />

        </LinearLayout>

        <Button
            android:id="@+id/btnUpload"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="9dp"
            android:text="Upload Document" />
    </LinearLayout>

```

```
        </LinearLayout>
    </RelativeLayout>
```

API.java

```
package com.example.uploadingpddffilestoserver;

import retrofit2.Call;
import retrofit2.http.Field;
import retrofit2.http.FormUrlEncoded;
import retrofit2.http.POST;

public interface Api {

    @FormUrlEncoded
    @POST("upload_document.php")
    Call<ResponsePOJO> uploadDocument(
        @Field("PDF") String encodedPDF
    );

}
```

ResponsePOJO.java :

```
package com.example.uploadingpddffilestoserver;

public class ResponsePOJO {

    private boolean status;
    private String remarks;

    public boolean isStatus() {

        return status;
    }

    public String getRemarks() {
        return remarks;
    }

}
```

Retrofitclient.java:

```
package com.example.uploadingpddffilestoserver;

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;
```

```

public class RetrofitClient {

    private static final String
BASE_URL="http://192.168.1.7/Android%20Tutorials/";
    private static RetrofitClient myClient;
    private Retrofit retrofit;

    private RetrofitClient() {
        retrofit = new
Retrofit.Builder().baseUrl(BASE_URL).addConverterFactory(GsonConverterFactory
.create()).build();
    }

    public static synchronized RetrofitClient getInstance() {
        if (myClient == null) {
            myClient = new RetrofitClient();
        }
        return myClient;
    }

    public Api getAPI() {
        return retrofit.create(Api.class);
    }

}

```

Code For uploading image:

MainActivity2.java

```

package com.example.uploadingpddffilestoserver;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Base64;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

import java.io.ByteArrayOutputStream;
import java.io.IOException;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class MainActivity2 extends AppCompatActivity {

    int IMG_REQUEST = 21;

```

```

        Bitmap bitmap;
        ImageView imageView;
        Button btnSelectImage, btnUploadImage;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            imageView = findViewById(R.id.imageView);
            btnSelectImage = findViewById(R.id.btnSelectImage);
            btnUploadImage = findViewById(R.id.btnUploadImage);

            btnSelectImage.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    Intent intent = new Intent();
                    intent.setType("image/*");
                    intent.setAction(Intent.ACTION_GET_CONTENT);
                    startActivityForResult(intent, IMG_REQUEST);

                }
            });

            btnUploadImage.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    uploadImage();
                }
            });

        }

        @Override
        protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
            super.onActivityResult(requestCode, resultCode, data);

            if(requestCode == IMG_REQUEST && resultCode == RESULT_OK && data !=
null){

                Uri path = data.getData();

                try {
                    bitmap =
MediaStore.Images.Media.getBitmap(getContentResolver(),path);
                    imageView.setImageBitmap(bitmap);

                } catch (IOException e) {
                    e.printStackTrace();
                }

            }

        }

        private void uploadImage() {

            ByteArrayOutputStream byteArrayOutputStream = new

```

```

ByteArrayOutputStream();
        bitmap.compress(Bitmap.CompressFormat.JPEG, 75,
byteArrayOutputStream);
        byte[] imageInByte = byteArrayOutputStream.toByteArray();
        String encodedImage =
Base64.encodeToString(imageInByte, Base64.DEFAULT);

        Call<ResponseKOJO> call =
RetroClient.getInstance().getAP().uploadImage(encodedImage);
        call.enqueue(new Callback<ResponseKOJO>() {
            @Override
            public void onResponse(Call<ResponseKOJO> call,
Response<ResponseKOJO> response) {
                Toast.makeText(MainActivity2.this,
response.body().getRemarks(), Toast.LENGTH_SHORT).show();

                if(response.body().isStatus()){

                }else{

                }

            }

            @Override
            public void onFailure(Call<ResponseKOJO> call, Throwable t) {
                Toast.makeText(MainActivity2.this, "image uploaded",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}

```

mainactivity2.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_horizontal"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="300dp"
            android:layout_height="300dp"
            android:layout_marginTop="33dp"
            tools:srcCompat="@tools:sample/backgrounds/scenic" />

        <Button
            android:id="@+id/btnSelectImage"

```

```

        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:text="Select Image" />

        <Button
            android:id="@+id/btnUploadImage"
            android:layout_width="300dp"
            android:layout_height="wrap_content"
            android:text="Upload Image" />

    </LinearLayout>
</RelativeLayout>

```

AP.java

```

package com.example.uploadingpddffilestoserver;

import retrofit2.Call;
import retrofit2.http.Field;
import retrofit2.http.FormUrlEncoded;
import retrofit2.http.POST;

public interface AP {
    @FormUrlEncoded
    @POST("upload_image.php")
    Call<ResponseKOJO> uploadImage(
        @Field("EN_IMAGE") String encodedImage
    );
}

```

ResponseKojo.java:

```

package com.example.uploadingpddffilestoserver;

public class ResponseKOJO {
    private boolean status;
    private String remarks;

    public boolean isStatus() {
        return status;
    }

    public String getRemarks() {
        return remarks;
    }
}

```

Retro client.java

```

package com.example.uploadingpddffilestoserver;

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class RetroClient {

```

```

    private static final String
BASE_URL="http://192.168.1.7/Android%20Tutorials/";
    private static RetroClient myClient;
    private Retrofit retrofit;

    private RetroClient(){
        retrofit=new
Retrofit.Builder().baseUrl(BASE_URL).addConverterFactory(GsonConverterFactory
.create()).build();
    }

    public static synchronized RetroClient getInstance(){
        if (myClient==null){
            myClient=new RetroClient();
        }
        return myClient;
    }

    public AP getAP(){
        return retrofit.create(AP.class);
    }

}

```

Upload document.php

```

<?php

include 'db_config.php';

$con = mysqli_connect($HOST, $USER, $PASSWORD, $DB_NAME);

$encodedPDF = $_POST['PDF'];
$command_exec = escapeshellcmd('python
C:\Users\MOHAN\Desktop\table-extraction\extract-invoice-data-master\main.py');
$str_output = shell_exec($command_exec);
echo $str_output;

$pdfTitle = "myPDF";
$pdfLocation = "documents/$pdfTitle.pdf";

$sqlQuery = "INSERT INTO `documents`(`title`, `location`) VALUES
('$pdfTitle', '$pdfLocation')";

if(mysqli_query($con, $sqlQuery)){

    file_put_contents($pdfLocation, base64_decode($encodedPDF));

    $result["status"] = TRUE;
    $result["remarks"] = "document uploaded successfully";
}

```

```

    }else{

        $result["status"] = FALSE;
        $result["remarks"] = "document uploading Failed";

    }

    mysqli_close($con);

    print(json_encode($result));

?>

```

Uploadimage.php

```

<?php

    include 'db_config.php';

    $con = mysqli_connect($HOST, $USER, $PASSWORD, $DB_NAME);

    $encodedImage = $_POST['EN_IMAGE'];
    $command_exec = escapeshellcmd('python
C:\Users\MOHAN\Desktop\table-extraction\extract-invoice-data-master\main.py');
    $str_output = shell_exec($command_exec);
    echo $str_output;

    $imageTitle = "myImage";
    $imageLocation = "documents/$imageTitle.jpg";

    $sqlQuery = "INSERT INTO `documents`(`title`, `location`) VALUES
('$imageTitle', '$imageLocation')";

    if(mysqli_query($con, $sqlQuery)){

        file_put_contents($imageLocation, base64_decode($encodedImage));

        $result["status"] = TRUE;
        $result["remarks"] = "Image Uploaded Successfully";

    }else{

        $result["status"] = FALSE;
        $result["remarks"] = "Image Uploading Failed";

    }

    mysqli_close($con);

    print(json_encode($result));

?>

```


Main.py

```
from tesseract_ocr import ocr
from preprocess import *
from pngcon import convo
from identify_company import identify_company
import numpy as np
import cv2
import time
from pathlib import Path

start_time = time.time()
file = r"C:\xampp\htdocs\Android Tutorials\documents\myPDF.pdf"

if (Path(file).suffix == '.pdf'):
    img = convo(file)
    opencvImage = cv2.cvtColor(np.array(img), cv2.COLOR_RGB2BGR)
else:
    opencvImage = cv2.imread(file)
print("Image conversion successful")

# Getting the ocr
txt = ocr(blackandwhite(opencvImage))
print("ocrd successfully")

identify_company(txt=txt)
print("--- %s seconds ---" % (time.time() - start_time))
```

Pngcon.py

```
"""
This module is for converting the pdf to png and handle any rotation detection
"""

# This method is used to convert the pdf
def convo(path):
    from pdf2image import convert_from_path

    images = convert_from_path(path, dpi= 500, poppler_path=r'C:\Program
Files\poppler-21.03.0\Library\bin')
    #images[0].save("sample.png", format = "PNG")

    return images[0]
```

Tesseract_ocr.py

```
'''
This is to detect and return the text
'''
def ocr(image):

    import pytesseract
    # Insert your pytesseract location here after installing
    pytesseract.tesseract_cmd = r'C:\Program
Files\Tesseract-OCR\tesseract.exe'

    data = pytesseract.image_to_string(image, lang='eng',config='--psm 6')
    ##print(data)
    return (data)
```

Preprocess.py

```
'''
This contains methods to preprocess the images
Dilated image = no noise + thick font
eroded image = no noise +thin font
'''

import cv2
import numpy as np
from matplotlib import pyplot as plt

# Converting to inverted image
def inverted(image):

    inverted_image = cv2.bitwise_not(image)
    #cv2.imshow(image)
    #cv2.imwrite('inverted.png',inverted_image)
    return cv2.bitwise_not(image)

# Converting to gray scale
def grayscale(image):

    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    #cv2.imshow(image)
    #cv2.imwrite('grayscale.png',image)
    return image
```

```

def canny(image):
    image = cv2.imread('jain.jpeg',0)
    edges = cv2.Canny(image,100,200)
    plt.subplot(121),plt.imshow(image,cmap = 'gray')
    plt.title('Original Image'), plt.xticks([]), plt.yticks([])
    plt.subplot(122),plt.imshow(edges,cmap = 'gray')
    plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
    # plt.show()
    # cv2.imwrite('canny.png',image)
    return image

# Converting to black and white
def blackandwhite(image):
    image = np.array(image)
    thresh, im_bw = cv2.threshold(grayScale(image), 200, 230,
cv2.THRESH_BINARY)

    #cv2.imshow("black and white",im_bw)
    #cv2.imwrite('bw.png',im_bw)
    #cv2.waitKey(0)
    return im_bw

# Removing noise
def noise_removal(image):
    import numpy as np
    kernal = np.ones((1,1), np.uint8)
    image = cv2.dilate(image, kernal, iterations=1)
    kernal = np.ones((1,1), np.uint8)
    image = cv2.erode(image, kernal, iterations=1)
    image = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernal)
    image = cv2.medianBlur(image, 3)

    #cv2.imshow(image)
    #cv2.imwrite('noise_removed.png',image)

    return (image)

# Converting to thin font
def thin_font(image):
    import numpy as np
    image = cv2.bitwise_not(image)
    kernal = np.ones((2,2), np.uint8)
    image = cv2.erode(image, kernal, iterations=1)
    image = cv2.bitwise_not(image)

    #cv2.imshow(image)
    cv2.imwrite('thin_font.png',image)

    return (image)

# Converting to thick font
def thick_font(image):

```

```

import numpy as np
image = cv2.bitwise_not(image)
kernal = np.ones((2,2), np.uint8)
image = cv2.dilate(image, kernal, iterations=1)
image = cv2.bitwise_not(image)

#cv2.imshow(image)
cv2.imwrite('thick_font.png',image)

return (image)

```

Identify_company.py

```

'''
Method to identify the company and call the needed module
'''

def identify_company(txt):

    from fuzzywuzzy import fuzz
    from jain import arrange_dump
    from unitron import unitron
    from table_extract import extract
    from sherays import Sheryas

    list_of_companies = ['SINo Particular Batch Expiry Date HSN/SAC Actual Qty
Billed Qty Rate Discount Amount','Description of Goods HSN/SAC Quantity Rate
per Amount',
'MKTD NO. RATE AMOUNT']

    lst = []
    for i in list_of_companies:
        lst.append(fuzz.token_set_ratio(i.lower(),txt.lower()))

    i = lst.index(max(lst))

    table = extract(txt)
    print("Table extracted")

    if i == 0:
        table.pop(0)
        #print(table)
        unitron(table)

    elif i == 1:
        table.pop(0)
        arrange_dump(table)

    elif i == 2:

```

```

        table.pop(0)
        table.pop(1)
        Sheryas(table)

    return

```

Table_extract.py

```

'''
This module is used to clean the text and extract the table
'''

from os import replace

def Clean_Text(txt):

    import re
    output = re.sub(r'^ \nA-Za-z0-9./]+', '', txt)

    return output.split('\n')

def find_heading(clean):
    from fuzzywuzzy import fuzz
    to_find = 'SINO SIND Description Particular HSN Rate Amount'
    starting = 0
    for t in clean:
        if (fuzz.token_set_ratio(to_find.lower(),t.lower())) > 70:
            starting = clean.index(t)

    return starting

def find_ending_serial(txt,loc):

    prev = 0
    t = ''

    for i in range(loc+1, len(txt)-1):
        temp = txt[i]

        if temp[0].isdigit() and int(temp[0]) == prev+1:

            #print('item detected',temp)
            prev = int(temp[0])
            t = temp
        else:

```

```

        continue
    return (txt.index(t))

def find_ending(txt):
    from fuzzywuzzy import fuzz

    to_find = ['Recieved', 'output', 'FLASH']
    lst = []
    highestacc = -1
    index = -1

    for i in to_find:
        for t in txt:
            lst.append(fuzz.token_set_ratio(i.lower(), t.lower()))

            if highestacc < max(lst):

                index = lst.index(max(lst))
                highestacc = max(lst)
                lst = []

            else:
                lst = []
                continue

    return index

def extract(txt):

    clean = Clean_Text(txt)
    print(clean)

    starting = find_heading(clean)
    ending = -1
    if 'SINo' in clean[starting] or 'SIND' in clean[starting] :
        #print('SERIAL EXECUTED')
        ending = find_ending_serial(clean, starting) + 2

    else:
        #print('THE OTHER ONE EXECUTED')
        ending = find_ending(clean)

    #print(starting)
    #print(ending)

    # Extracting the rows of the table
    op = []

    for i in range(starting, ending-1):

```

```

        op.append(clean[i])
    return op

```

jain.py

```

'''
Rakesh
This module is to convert the rows of Jain Chemicals into a table and store
the json file
'''

'''From Reverse:
1 ---> n=len(headers) ---> 6

num[0] ---> n ---> headers[n-1]
num[1] ---> n-1 ---> headers[n-1-1]
num[2] ---> n-2 ---> headers[n-2-1]
num[3] ---> n-3 ---> headers[n-3-1]
num[4] ---> If it is a number then only insert or else leave it ---> n-4
num[4] ---> Concatenate the left out list elements and rev them ---> n-5
--->headers[n-5-1]

'''
import csv

def arrange_dump(contents):

    import json
    import copy
    from utilites import rev_sentence,listToString
    u=0
    # Intializing the needed elements
    heads = ['Description of Goods', 'HSN/SAC', 'Quantity', 'Rate', 'per',
'Amount']
    key_list = heads
    field_names = key_list
    headers = copy.deepcopy(headers)
    final = []
    file = open('Jain Chemicals.json','a')

    # Reverse sorting the list
    for i in contents:
        # Separating to individual contents
        temp = i.split(' ')
        temp = ' '.join(temp).split()
        #print(temp)
        if len(temp) == 0:
            continue

```

```

# Merging the quantity column
try:
    #print(temp.index('Nos'))
    index_pos = temp.index('Nos')
except ValueError:
    #print(temp.index('KGS'))
    index_pos = temp.index('KGS')
k=(temp[index_pos-1]+temp[index_pos])
# print(k)
n=index_pos
del temp[n-1:n+1]
temp.insert(n-1,k)
#print(temp)
temp.reverse()
#Storing the reversed list
final.append(temp)

# Arrainging in cloumns and storing to file
for st in final:

    n=len(headers)
    m=len(st)
    for i in range(n):
        if i==0:
            headers[n-1]=st[i]
        elif i==4:
            if st[i].isdigit():
                headers[n-i-1] = st[i]
                a=i+1
            else:
                headers[n-i-1] = " "
                a=i

        elif i==n-1:
            c=[]
            for i in range(a,m):
                c.append(st[i])
            y=listToString(c)
            w=rev_sentence(y)
            headers[0]=w

        else:
            headers[n-i-1] = st[i]

    dict_from_list = dict(zip(key_list, headers))
    cars=[]
    cp=dict_from_list.copy()
    cars.append(cp)

```



```
with open('Names.csv', 'a') as csvfile:

    if u==0:
        writer = csv.DictWriter(csvfile, fieldnames = field_names)
        writer.writeheader()
        writer.writerows(cars)
        u=u+1
    else:
        writer = csv.DictWriter(csvfile, fieldnames = field_names)
        writer.writerows(cars)

    json.dump(dict_from_list,file,indent=4)

file.close()
print("Names.csv")
```