CHAPTER 1

INTRODUCTION

1.1 Overview

This study looks for a technique that may be used to extract any data from regularly used printed bills and PDF invoices. Later on, the information from these bills or invoices can be extensively utilised, for example, in statistical or machine learning applications. This study focuses on the extraction of final bill-amount, itinerary, date, and related information from bills and invoices because they include a wealth of information about the users' purchases, preferences, and other activities. With the help of optical character recognition (OCR) technology, printed or handwritten characters can be fully recognised from photographs. Initially, OpenCV was used to remove any extraneous noise from the image while still detecting the bill or invoice in it. Next, the intermediate image is transmitted.

1.2 Objectives

- To accomplish the tasks by imitating the human actions that include processing an invoice by an individual.
- To minimize the workflow of an employee.
- To build a software application that automates the process.
- To design a simple UI which takes input to be processed and gives the output.
- To implement the automation of invoice processing using tools like pdfplumber, invoice2data, tesseract etc.

1.3 Purpose, scope, and applicability

1.3.1 Purpose

To extract pertinent information from bills, such as the vendor’s name, invoice date, invoice number, line items, and total amount, pdfplumber uses an OCR (Optical Character Recognition) approach. Utilising OCR technology, it is feasible to electronically extract data from an invoice by having the software recognise the letters and text within the invoice image.

A Python package called pdfplumber offers an OCR-based method for extracting data from PDF documents. It enables users to extract data from particular regions of a PDF document as well as text, tables, and images from PDF files. Users may automate the workflow for processing invoices with pdfplumber, which lowers the amount of manual labour needed to extract data from invoices.

1.3.2 Scope

Automation of Accounts Payable: Pdfplumber can automate the workflow for processing invoices, making it simpler for businesses to handle their accounts payable procedure. For businesses who often get a lot of invoices, this can be especially helpful.

Healthcare Industry: Because of the wide range of services provided and the numerous payers involved, invoice processing in the healthcare sector may be rather complicated. By automating the data extraction and lowering the amount of manual labour necessary, Pdfplumber can assist in streamlining the procedure.

Retail Sector: Businesses in the retail sector frequently handle a lot of bills from suppliers. Automating the processing of invoices can help to minimise errors and guarantee on-time payments.

Legal Sector: Court reporters, expert witnesses, and other service providers frequently submit invoices to law firms. The processing of invoices can be made more efficient with Pdfplumber, and the time and materials needed for manual data entry can be decreased.

1.3.3 Applicability

Data Extraction from Invoices: Pdfplumber can be used to extract pertinent information from invoices, such as the name of the vendor, the date and number of the invoice, the line items, and the total amount. The utilisation of this information then allows for further processing or system integration.

Verifying an invoice's accuracy involves comparing the data to the original invoice document using a programme called Pdfplumber. By doing so, errors can be decreased and fraud can be stopped.

Tracking of bills is possible with Pdfplumber, including monitoring their approval, processing, and payment statuses. This can make it more likely that payments will be made on time and that invoices will be processed on schedule.

Pdfplumber can be used to electronically archive invoices, making it simpler to recover them at a later time. This could lessen the need for physical storage space and enhance record-keeping.

Analysis of Invoice Data: Pdfplumber can be used to analyse invoice data in order to spot trends, monitor spending habits, and spot areas where money can be saved. This can aid companies in increasing their financial performance and optimising their expenses.

1.4 Organization of report

The rest of the report is divided into 4 chapters as follows. Chapter 2 deals with the review of related works for our project. Chapter 3 describes the requirements for the project. This chapter describes both software and hardware requirements. Chapter 4 describes the actual plan of execution for developing our project and finally Chapter 5 gives the system design of our project. Chapter 6 is implementation, Chapter 7 is Testing, Chapter 8 is result and performance analysis and finally Chapter 9 gives the application, conclusion and future development aspects of our project.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

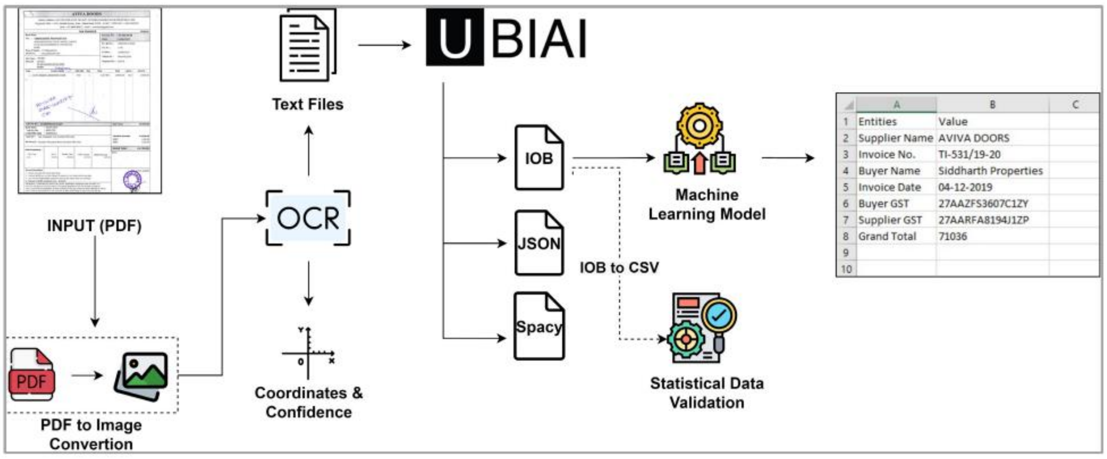
Literature survey or Literature Review is a survey of previously existing scholarly resources such as books, journals, and articles related to specific topics or questions. It involves the search and evaluation of available literature in your given subject or chosen topic area. It documents the state of the art concerning the subject or topic you are writing about. Concerning the project, a literature survey was conducted to get a better picture of the idea, for idea development to understand the methodology that is currently being used to learn about the limitations present in the correct methodologies.

2.2 Summary of Papers

The below section provides a brief summary of the reference paper utilized for Invoice processing using OCR and deep learning.

Dipali Baviskar et al.,[1] The daily transaction of an organization generates a vast amount of unstructured data such as invoices and purchase orders. Managing and analysing/ unstructured data is a costly affair for the organization. Unstructured data has a wealth of hidden valuable information. Extracting such insights automatically from unstructured documents can significantly increase the productivity of an organization. Thus, there is a huge demand to develop a tool that can automate the extraction of key fields from unstructured documents. Researchers have used different approaches for extracting key fields, but the lack of annotated and high-quality datasets is the biggest challenge. Existing work in this area has used standard and custom datasets for extracting key fields from unstructured documents. Still, the existing datasets face some serious challenges, such as poor-quality images, domain-related datasets, and a lack of data validation approaches to evaluate data quality. This work highlights the detailed process flow for end-to-end key fields extraction from unstructured documents. This work presents a high-quality, multi-layout unstructured invoice documents dataset assessed with a statistical data validation technique. The proposed multi-layout unstructured invoice documents dataset is highly diverse in invoice layouts to generalize key field extraction tasks for unstructured documents. The proposed multi-layout unstructured invoice documents dataset is evaluated with various feature extraction techniques such as

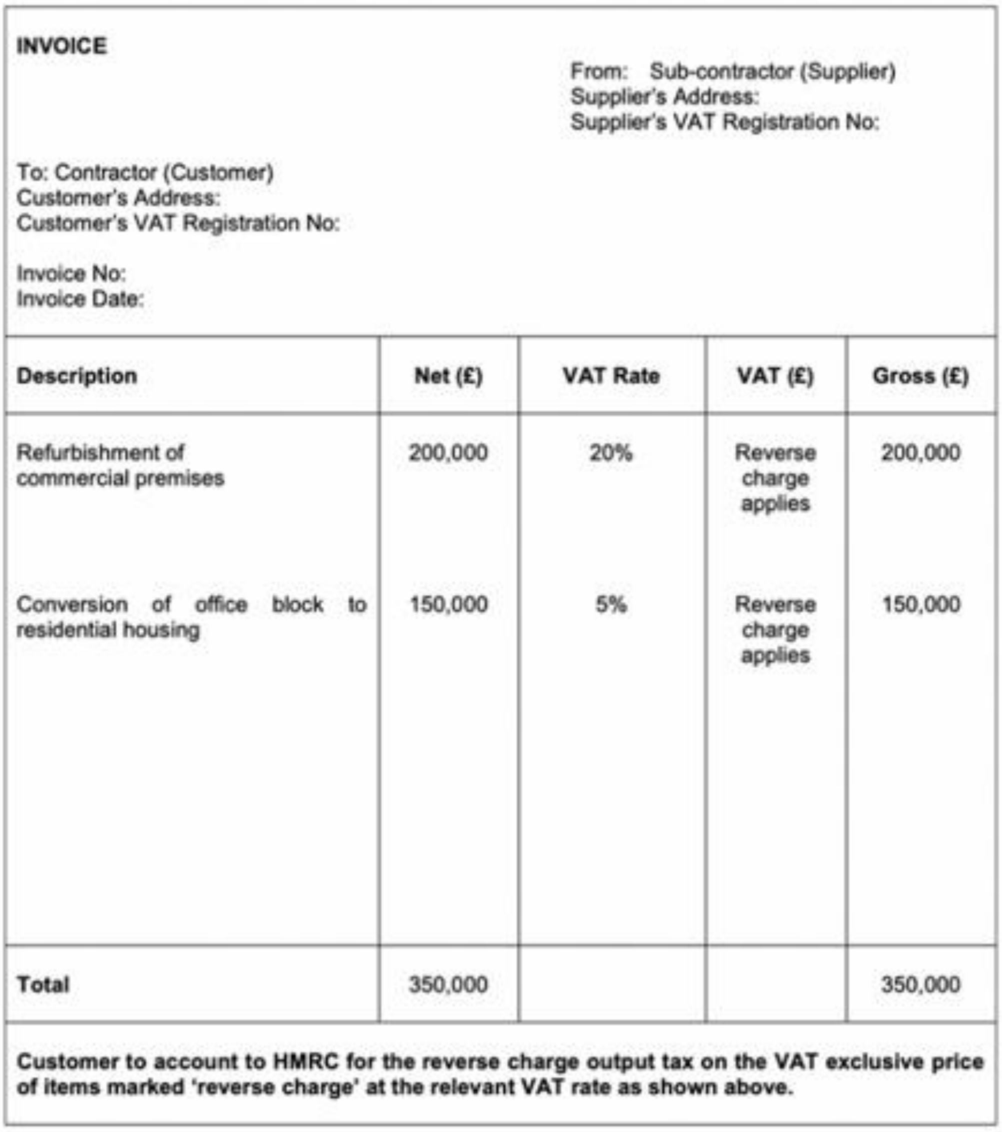
Glove, Word2Vec, FastText, and AI approaches such as BiLSTM and BiLSTM-CRF. We also present the comparative analysis of feature extraction techniques and AI approaches on the proposed multi-layout unstructured invoice document dataset. We attained the best results with BiLSTM-CRFmodel.



	A	B	C
1	Entities	Value	
2	Supplier Name	AVIVA DOORS	
3	Invoice No.	TI-531/19-20	
4	Buyer Name	Siddharth Properties	
5	Invoice Date	04-12-2019	
6	Buyer GST	27AAZFS3607C1ZY	
7	Supplier GST	27AARFA8194J1ZP	
8	Grand Total	71036	
9			
10			

Figure 2.2.1: Detailed Process Flow for Key Fields Extraction from Multi-layout Unstructured Invoice Documents.

David A et al.,[2] This paper propose a method of invoice document structure analysis that provides a means to extract the relevant information from an unknown invoice. Our method uses a combination of textual and graphical processing by analysing the line and line intersection features in the document as well LIS searching for possible keywords such as item number; quantity, total, etc. Valid keyword search regions are determined by a specialized connected-component analysis before any OCR is performed. The results of the keyword search and the line analysis are combined to give the search regions for extracting the relevant data contained in the invoice. This analysis will become pan of a larger invoice interpretation system which is currently under development. In addition to the textual and graphical processing techniques, our method also incorporates advanced OCR (Optical Character Recognition) technology to accurately extract text from the invoice document. By leveraging OCR, we can convert the scanned or image-based text into machine-readable format, enabling precise analysis and extraction of the relevant information. By combining the results of keyword search, line analysis, and OCR, we are able to determine the optimal search regions for extracting the required data from the invoice. This comprehensive analysis approach improves the reliability and accuracy of the extracted information, ensuring that the relevant invoice details are captured effectively.



INVOICE

From: Sub-contractor (Supplier)
Supplier's Address:
Supplier's VAT Registration No:

To: Contractor (Customer)
Customer's Address:
Customer's VAT Registration No:

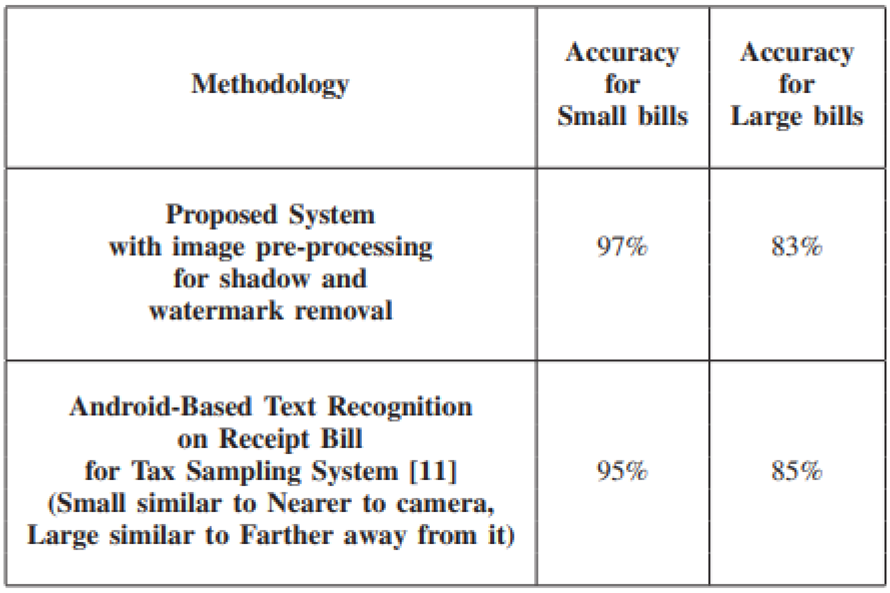
Invoice No:
Invoice Date:

Description	Net (£)	VAT Rate	VAT (£)	Gross (£)
Refurbishment of commercial premises	200,000	20%	Reverse charge applies	200,000
Conversion of office block to residential housing	150,000	5%	Reverse charge applies	150,000
Total	350,000			350,000

Customer to account to HMRC for the reverse charge output tax on the VAT exclusive price of items marked 'reverse charge' at the relevant VAT rate as shown above.

Figure 2.2.2**:** A sample invoice document containing keyword headings and enclosing boxes.

Vedant Kumar et al., [3] 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.



Methodology	Accuracy for Small bills	Accuracy for Large bills
Proposed System with image pre-processing for shadow and watermark removal	97%	83%
Android-Based Text Recognition on Receipt Bill for Tax Sampling System [11] (Small similar to Nearer to camera, Large similar to Farther away from it)	95%	85%

Figure 2.2.3: Results and Comparisons.

Najoua Rahal et al.,[4] The relevant entity extraction from scanned document image is a very challenging task due to highly heterogeneous templates, and several structure layouts. These problems lead to inaccuracy for document image recognized by OCR. In this paper, we propose an effective solution for these problems, in which the relevant entities are extracted from Arabic and Latin scanned invoices. The input of the system is an invoice image which is submitted to an OCR without layout analysis. After, invoices are labeled in the text recognized by the OCR. By combining the logical and physical structures, a local graph model is built for extraction entity. Finally, we implement a correction module which requires the mislabeling correction by eliminating the superfluous parts detected by labeling step. We evaluate the obtained results with 1050 real invoices as reported in experimental section. Additionally, our solution incorporates machine learning algorithms to improve the accuracy of entity extraction from scanned invoice images. By training the models on a large dataset of labeled invoices, the system can learn patterns and characteristics specific to Arabic and Latin invoices, enhancing its ability to accurately extract relevant entities. To ensure the correctness of the extracted entities, we have implemented a correction module that detects and rectifies mislabeling errors. This module identifies and eliminates any superfluous parts that may have been incorrectly labeled during the OCR and labeling steps, thus refining the accuracy of the extracted entities.

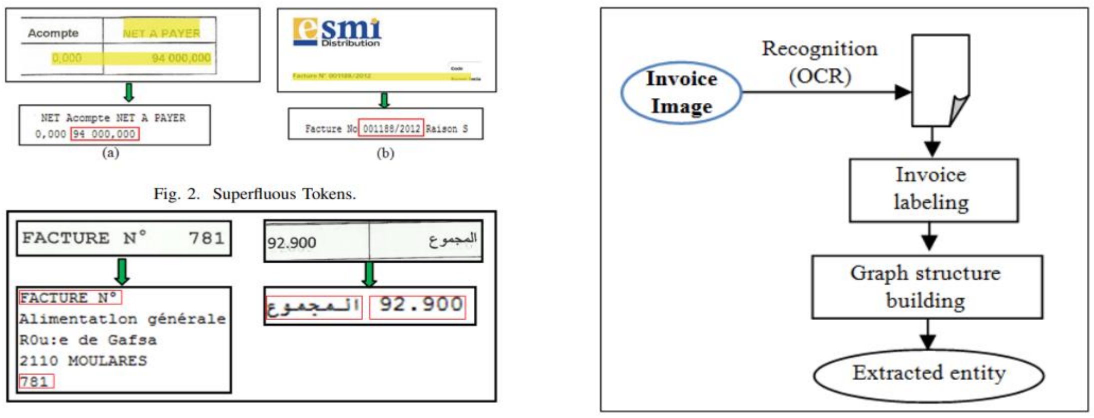


Figure 2.2.4: Proposed System

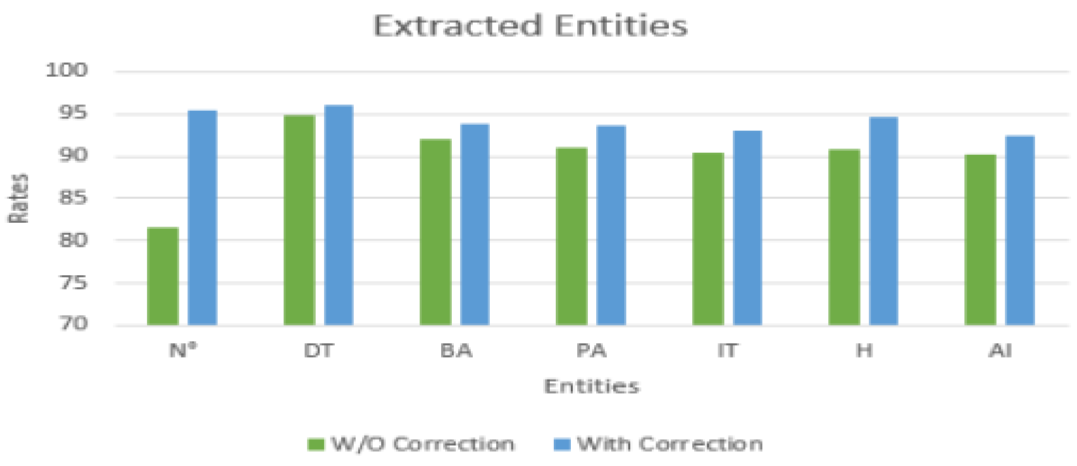


Figure 2.2.5: Extraction results of the entities

Md. Danish Quaum1 et al.,[5] have proposed, Invoice processing and mailing refers to the task that extracts data from various invoices and store it in a excel file, after that an email should automatically be sent to the mentioned email address. And to reduce human intervention in sensitive and repetitive task otherwise a small wrong entry may lead to consequences in business statistical results. Invoices maintaining is an essential segment of business for various purpose so this project can be extensive used to overcome human workforce handling this job so they can be used in other activities that require humanly decision. Additionally, implementing an automated invoice processing and mailing system offers several advantages. It not only reduces the chances of errors or inconsistencies in data entry but also enhances the overall efficiency and speed of the process. By automating these tasks, businesses can save valuable time and resources, allowing employees to focus on more strategic and value-added activities.

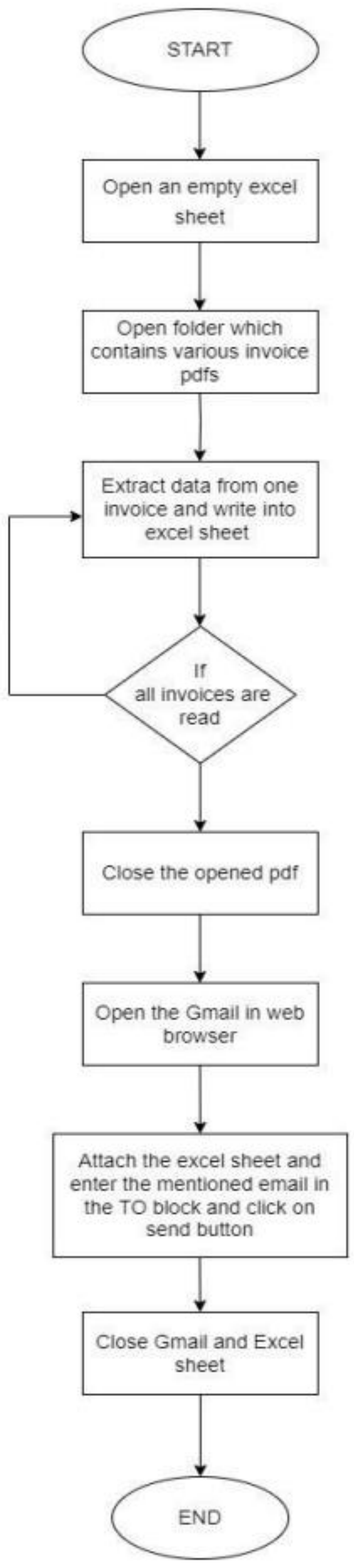

START
Open an empty excel sheet
Open folder which contains various invoice pdfs
Extract data from one invoice and write into excel sheet
If all invoices are read
Close the opened pdf
Open the Gmail in web browser
Attach the excel sheet and enter the mentioned email in the TO block and click on send button
Close Gmail and Excel sheet
END

Figure 2.2.6: Flowchart of proposed method.

Trapti Upadhyay et al., [6] This paper tells that Receipt preparing is one of the key back-office activities that requests a great deal of manual exertion. With manual cycles set up, organizations sit around idly acquiring, allotting and affirming Receipts. (RPA) Robotic Process Automation is a more effective approach to Process invoicing. In this project we are going to Extract Invoice data and then filter them according to their due date and put those invoices into their respective excel sheets. Further the Excel sheet with expired Due Date will be sent to Respective Email Address. This Invoice processing Automation System is ,also trained to understand real world documents. It can also identify and extract a wide range of business values from your receipts and invoices, depending on the needs and requirements of your accounts payable and expense compliance processes. This Automation System will Process Invoices in Less time and with no Errors. This Automation Sem can Process More than 100’s of Invoices Per day which is very hard to get processed manually. It will save Human Efforts, it will give the company a monetary gain with minimum errors. Additionally, the Robotic Process Automation (RPA) approach implemented in this project offers scalability and flexibility, allowing the system to handle a large volume of invoices efficiently. By automating the invoice data extraction and filtering process, the system eliminates the need for manual intervention, reducing the chances of errors and ensuring consistent and accurate results which can be seen.

Xufeng Ling et al., [7] RPA (robot process automation) is an intelligent tool that used to automatically process the business documents. Partially, incorporating human input into the learning process helps to improve the efficiency over 200% while the error could be controlled within 0.5%. To this effect, firstly, we applied OCR to the document (either PDF or picture) to identify the specific element; secondly, the text features of the document are extracted by natural language processing technology; and at the end, an automatic description and classification are proposed using machine learning algorithm. We also incorporated the human-computer interaction into the proposed system for modification and validation of the extracted features. Indeed, the proposed model transforms the experience of document administrators into artificial intelligence algorithm through RPA document processing, which greatly reduces the influence of leave and job transfer of the document administrators, and solves the about problem of document processing and meets the needs of the work.

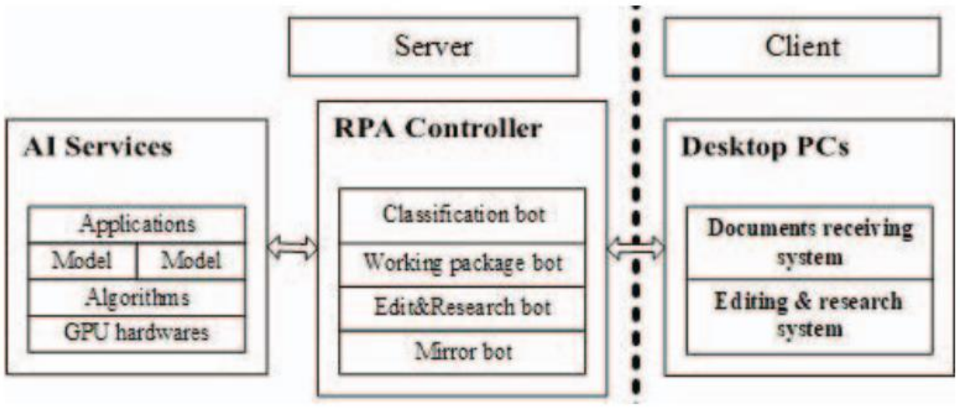
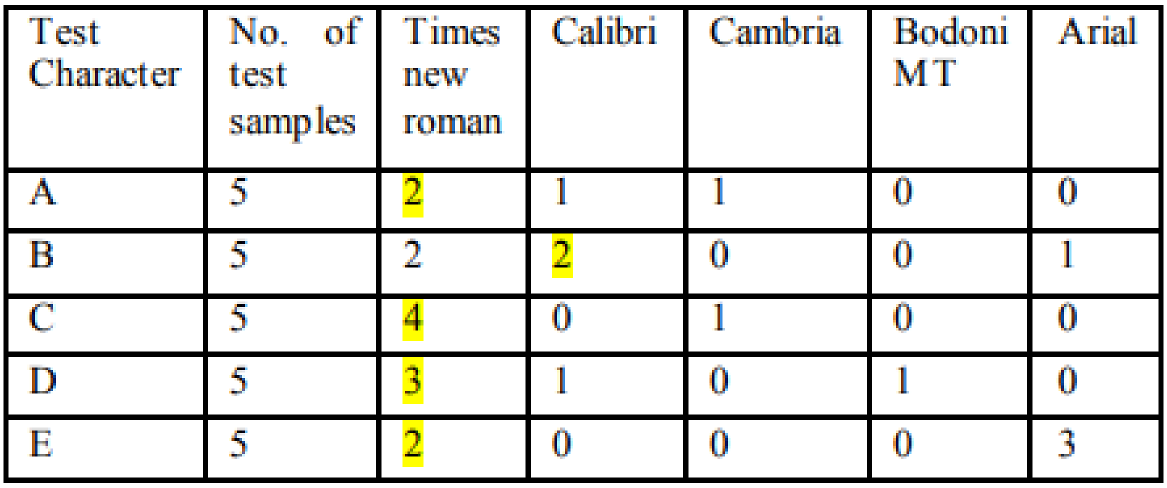


Figure 2.2.**7:** Frame diagram of RPA system

Bharath V et al., [8] The inclination of optical technologies like OCR lies in achieving higher recognition rates with optimal or reduced computational complexities. At present there exist optical technologies for automation of reading the text from document images with almost nearing to 100% accuracy. Especially, the Roman language OCR's are reliable and robust enough in producing higher accuracies by being able to recognize varying font styles of varying sizes. However for the font style/ size independent OCR's one of the main aspect is its computational complexity. It is significant concern to reduce the computational complexities involved in the process of character recognition through a font style / size independent OCR. In this paper, a technique for classification of the font style based on character image is proposed by employing the distance profile features with respect to left, right and diagonal directions of a character image. The major objective of this work is to reduce the complexity of the generic OCR systems by font style recognition. The di stance profile features of character images are fed to a support vector machine classifier. For experimentation, the training data sets are comprised of around 10 widely used font styles of upper case letters as well as lower case letters. The testing is conducted wi th the character images that are extracted from various non editable document sources comprising of 5 different font styles. The performance of algorithm is found to be satisfactory with an accuracy of 80%. Furthermore, the proposed technique of font style classification based on distance profile features can contribute to the overall improvement of font style-independent OCR systems. By accurately identifying the font style of a character image, the computational complexity of OCR processes can be significantly reduced, leading to faster and more efficient recognition



Test Character	No. of test samples	Times new roman	Calibri	Cambria	Bodoni MT	Arial
A	5	2	1	1	0	0
B	5	2	2	0	0	1
C	5	4	0	1	0	0
D	5	3	1	0	1	0
E	5	2	0	0	0	3

Figure 2.2.8: Confusion Matrix.

Rohit Saluja et al., [9] Conventional approaches to spell checking suggest spelling corrections using proximity-based matches to a known vocabulary. For highly inflectional Indian languages, any off-the shelf vocabulary is significantly incomplete, since a large fraction of words in Indic documents are generated using word conjoining rules. Therefore, a tremendous manual effort is needed in spell correcting words in Indic OCR documents. Moreover, in a spell-checking system, a vocabulary may suggest multiple alternatives to the incorrect word. The ranking of these corrective suggestions is improved using language models. Owing to corpus resource scarcity, however, Indian languages lack reliable language models. Thus, learning the character (or n-gram) confusions or error patterns of the OCR system can be helpful in correcting the Out of Vocabulary (OOV) words in OCR documents. We adopt a Long Short-Term Memory (LSTM) based character level language model with a fixed delay for discriminative language modeling in the context of OCR errors for jointly addressing the problems of error detection and correction in Indic OCR. For words that need not be corrected in the OCR output, our model simply abstains from suggesting any changes. We present extensive results to validate the performance of our model on four Indian languages with different inflectional complexities. We achieve F-Scores above 92.4% and decreases in Word Error Rates (WER) of at least 26.7% across the four languages. By leveraging the power of deep learning and character-level modeling, our system significantly reduces the manual effort required for spell checking Indic OCR documents. It overcomes the limitations of traditional proximity-based methods and incomplete vocabularies by learning the specific error patterns and confusions of the OCR system itself.

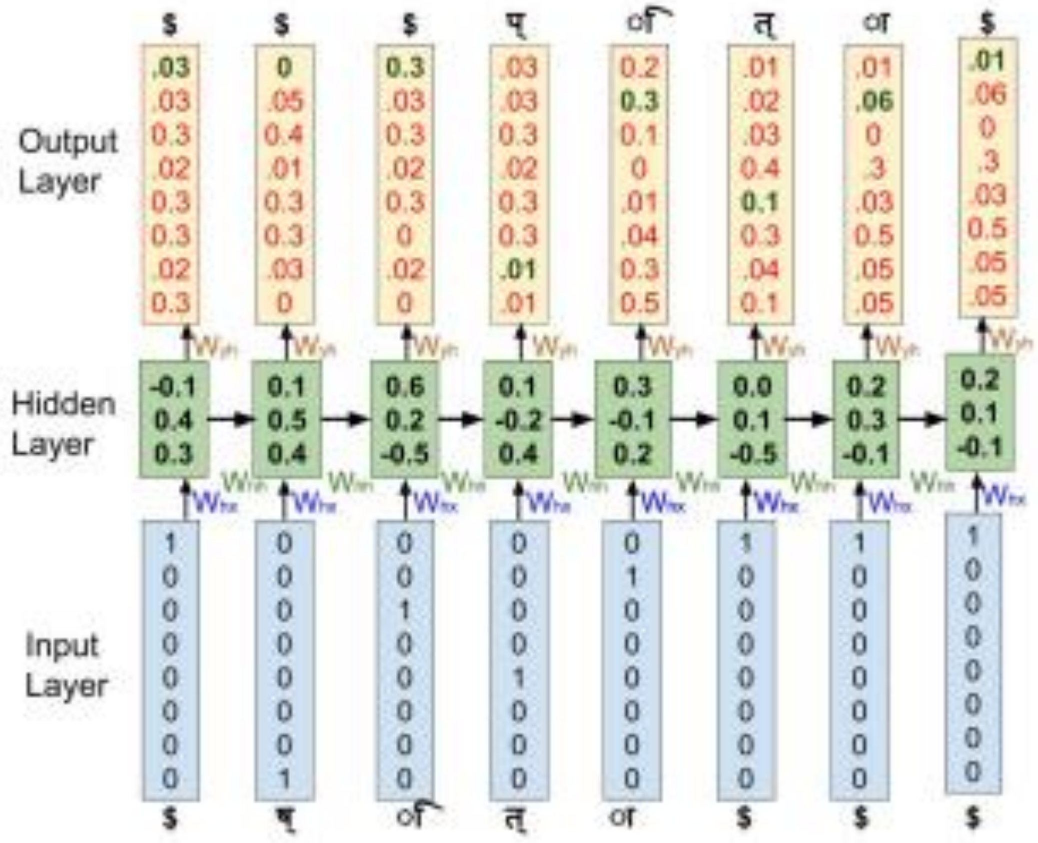


Figure 2.2.9**:** LSM model with 2 units delay.

Harshit Sidhwa et al., [10] 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 and it satisfies PDF also.

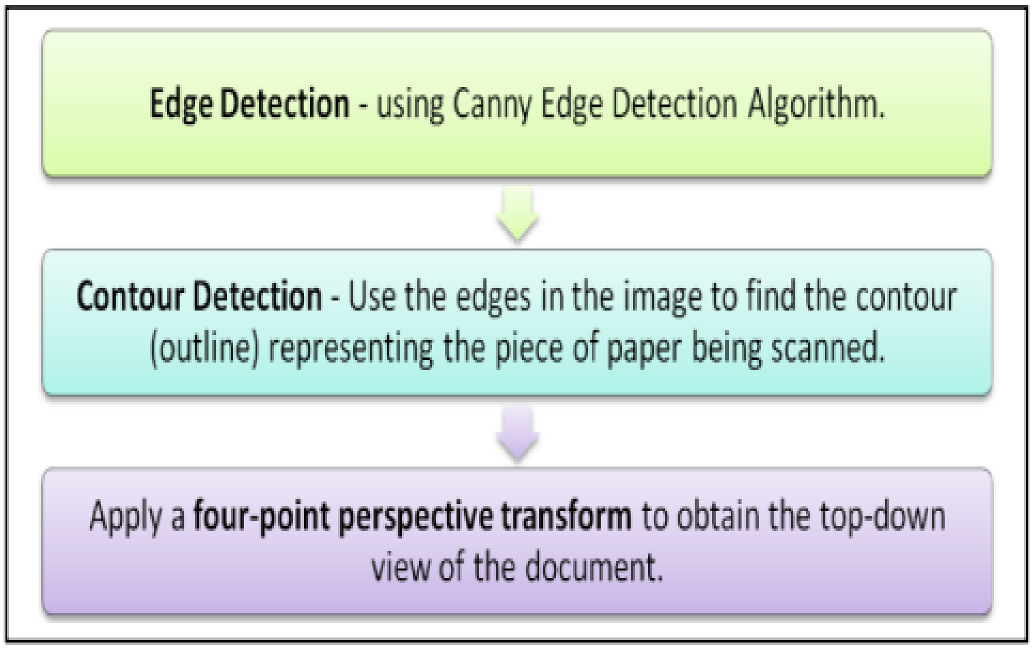

Edge Detection - using Canny Edge Detection Algorithm.
Contour Detection - Use the edges in the image to find the contour (outline) representing the piece of paper being scanned.
Apply a four-point perspective transform to obtain the top-down view of the document.

Figure 2.2.10: Outline of the methodology used in our research work

2.3 Drawbacks of Existing Systems

- The existing system of manual invoice processing is a time-consuming and error-prone process.
- It involves manual data entry, which is tedious and can result in errors.
- The manual process can cause delays in payment processing and can negatively affect vendor relationships.
- Additionally, manual invoice processing is not scalable and can become overwhelming for businesses with a high volume of invoices to process.

2.4 Problem Statement

An automated solution for accurate and efficient extraction of information from PDF invoices, reducing processing time”

Input: The input for the system is a PDF invoice containing information such as part, price, description, and amount and quantity.

Output: The output of the system is the extracted information from the input PDF invoice, stored in a usable format such as a database or spreadsheet.

2.5 Proposed Solution

The proposed system using pdfplumber to process invoices provides an automated and efficient solution for extracting and processing data from PDF invoices. With the use of Python programming and related software tools, the system can accurately extract key information such as part, price, description, and amount and quantity. The system can then store the extracted data in a usable format, such as a database or spreadsheet, for further analysis and reporting. The use of pdfplumber and other software tools significantly reduces the manual effort required to process invoices, resulting in increased efficiency and reduced processing time.

CHAPTER 3

REQUIREMENT ENGINEERING

In the engineering design process, identifying, documenting, and managing requirements is referred to as requirement engineering. It offers a method for comprehending the customer's needs, examining the situation and determining its viability, negotiating a fair solution, explicitly defining the solution, confirming the specifications, and managing the requirements as they are turned into a functional system.

The specifications for a hardware device are known as the hardware requirements. A software need is a condition or a capacity that a user needs in order to solve a problem or accomplish a goal. A need or capability that a system or system component must satisfy in order to comply with a contract, standard, specification, or other formally enforced document.

3.1 Software and Hardware Tools Used

The correct software tools must be chosen because some are superior to others in many ways. While they may all serve the same objective, some are more efficient and convenient than others. The following tools have been selected after thorough consideration of our needs and the best tools to suit those goals. A good programme should be resource-light or capable of running on all processors. The chosen tools should have a user-friendly interface and provide clear documentation or support resources to assist with any questions or issues that may arise during implementation.

3.1.1 Hardware Requirements

- Operating system: Windows, Mac
- RAM: Minimum 8GB or higher
- GPU: 4GB dedicated
- Processor: Intel Core i3 or higher
- HDD: 10GB or higher
- Monitor: 15” or 17” colour monitor

3.1.2 Software Requirements

- **Python**: pdfplumber and other data-processing programmes are created using the well-liked Python programming language.
- **pdfplumber library**: To extract information from PDF documents, use this Python package. The pdfplumber library needs to be set up in the Python environment before it can be used.
- **Integrated Development Environment (IDE):** An IDE is a piece of software that offers a setting for creating software. Code highlighting, debugging tools, and other helpful features are frequently included. Python applications can be created using a variety of IDEs, including PyCharm, Visual Studio Code, and Spyder.
- **Microsoft Excel** is a spreadsheet programme that may be used to store and examine data from extracted invoices.

3.2 Conceptual/ Analysis Modelling

3.2.1 Use case diagram

A use case diagram at its simplest is a representation of a User interaction with the system that shows the relationship between the User and the different use cases in which the user, Developer, client is involved as in Figure. The different interactions with system include pre-processing of input invoices, Detection of keywords such as part, description, total, quantity and price, classification of the Dataset based on the keywords, and after extracting the keyword the result is stored in the excel sheet. The use case diagram also showcases the flow of actions within the system. This includes the identification of relevant information from input invoices, such as vendor details, invoice numbers, and dates. The system further processes the extracted data to perform calculations, validations, and error handling, ensuring the accuracy and integrity of the information. The use case diagram demonstrates the system's capability to generate reports or notifications based on the processed data. This enables users, developers, and clients to access valuable insights, track invoice statuses, and make informed decisions. The diagram also reflects the integration of the system with external entities, such as databases or other systems, for seamless data exchange and synchronization.

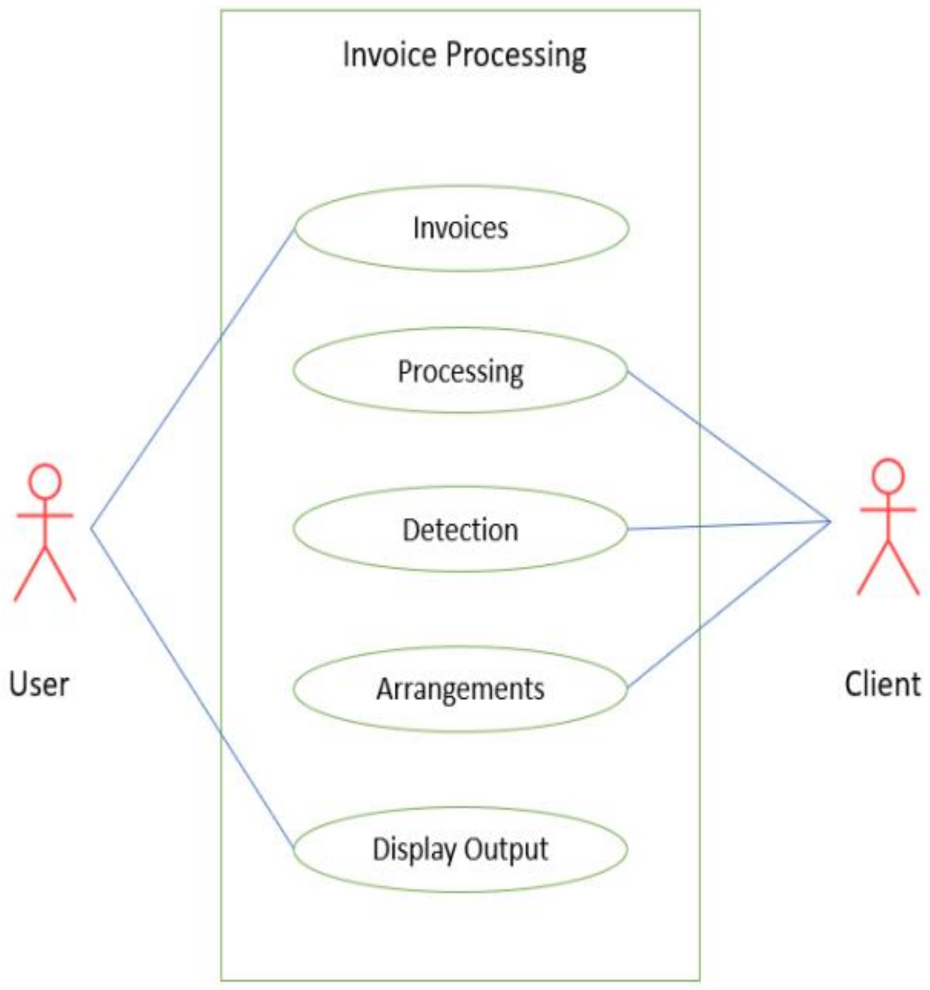


Figure 3.1: Use Case Diagram of Proposed Model

3.2.2 Sequence diagram

A sequence diagram shows object interactions arranged in time sequence. As shown below in Figure, the first step in sequence is to upload the invoices from local storage. This is followed by upload of the invoices to the system. After the successful upload of the invoice pre-processing of the invoice is carried out. The next successor step is to remove the spaces in the invoices, Next step is to extract the relevant information from the invoice. And the final step is displaying the result in the user interface of our system. Once the relevant information is extracted from the invoice, additional processing steps may be performed, such as data validation, formatting, or calculations. These steps ensure the accuracy and consistency of the extracted data

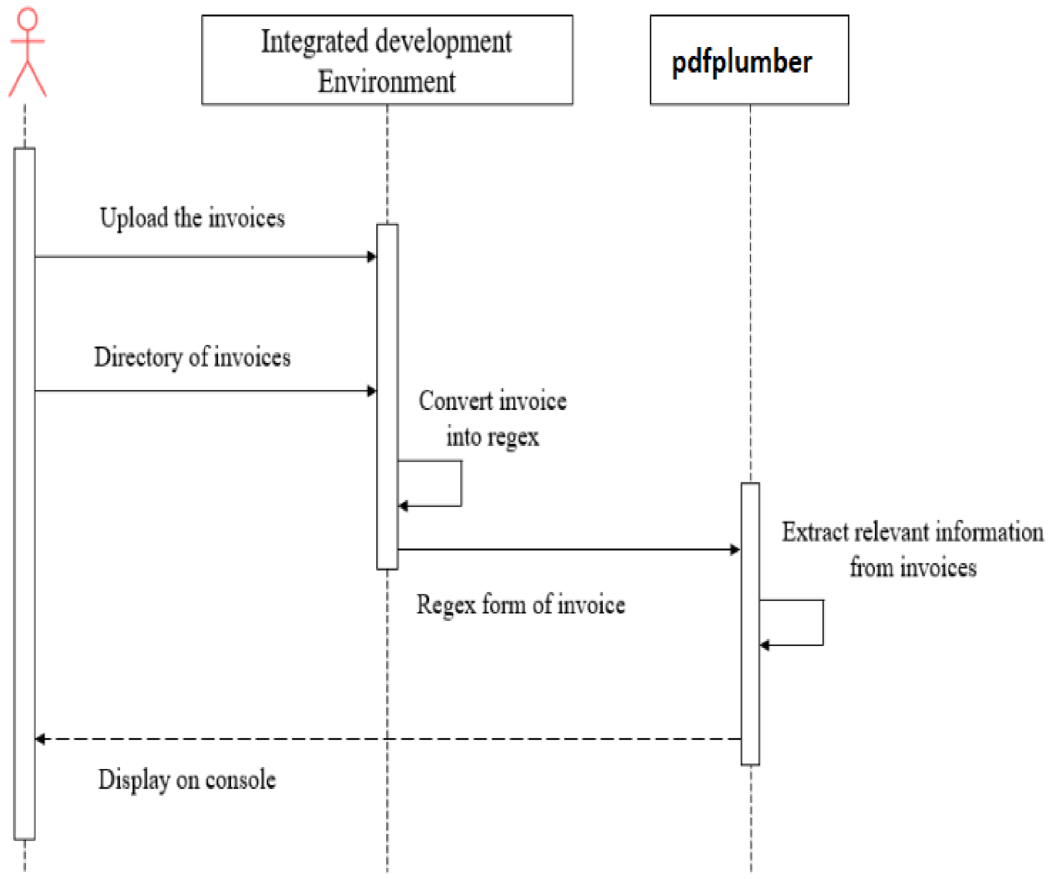

Integrated development Environment
pdfplumber
Upload the invoices
Directory of invoices
Convert invoice into regex
Extract relevant information from invoices
Regex form of invoice
Display on console

Figure 3.2: Sequence Diagram of Proposed Model

3.2.3 Activity diagram

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system as depicted in figure. The series of activities performed in invoice processing is shown below. It starts with the input of the invoice from local storage and Preprocessing techniques are applied. It is then followed by detection of relevant fields and then arrangements of invoice details and finally asking the user to continue to input other invoice or to exit the system. The activity diagram provides a visual representation of the flow of activities in the invoice processing system, highlighting the key steps involved in extracting and organizing invoice data. It serves as a valuable tool for understanding and communicating the system's behavior and can assist in the design, analysis, and optimization of the invoice processing workflow.

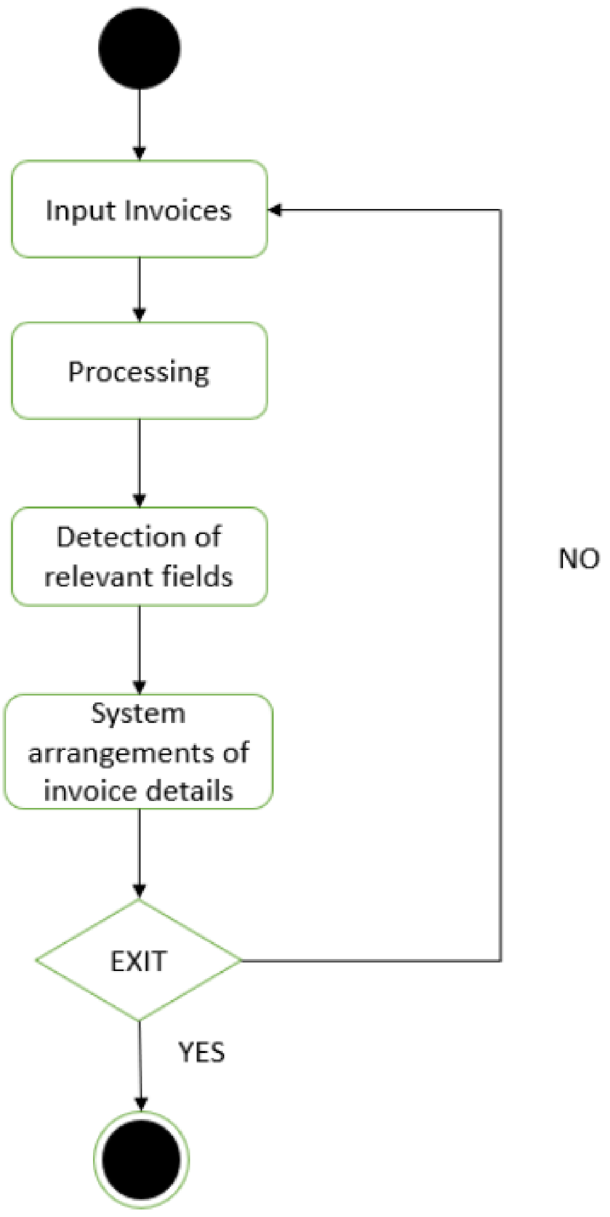

Input Invoices
Processing
Detection of relevant fields
System arrangements of invoice details
EXIT
NO
YES

Figure 3.3: Activity Diagram of Proposed Model

3.2.4 State chart Diagram

As shown in figure. The different states involved include Input Invoices, Pre-processing, Detection of keywords, Arrangement's invoice details and finally displaying the output on the console and the displayed the output in the excel sheet so that the excel can we used by the one who is in need.

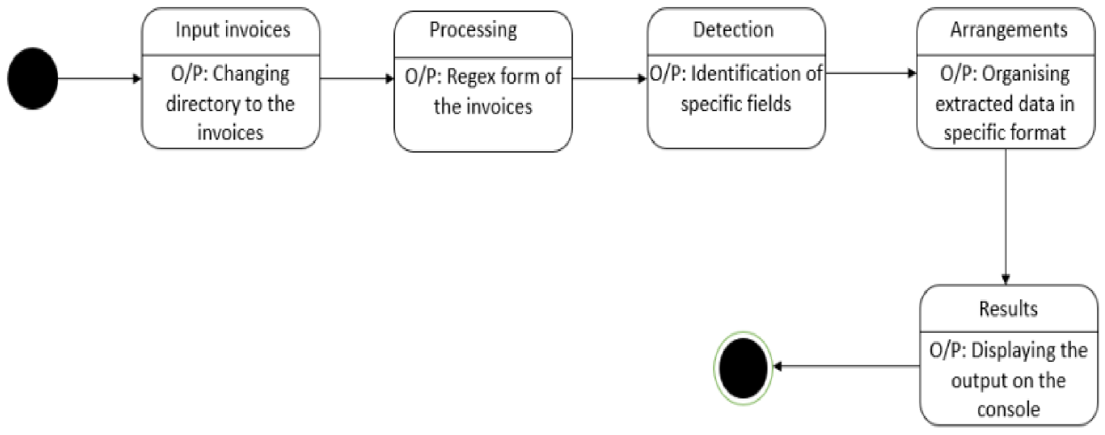

Input invoices
O/P: Changing directory to the invoices
Processing
O/P: Regex form of the invoices
Detection
O/P: Identification of specific fields
Arrangements
O/P: Organising extracted data in specific format
Results
O/P: Displaying the output on the console

Figure 3.4: State Chart Diagram of Proposed Model

3.2.5 Class Diagram

By displaying the classes and their connections to one another, a class diagram, a form of UML (Unified Modelling Language) diagram, illustrates the structure of a system or software application. It is a visual depiction of a system's classes, interfaces, and objects, as well as their properties, methods, and connections to one another.

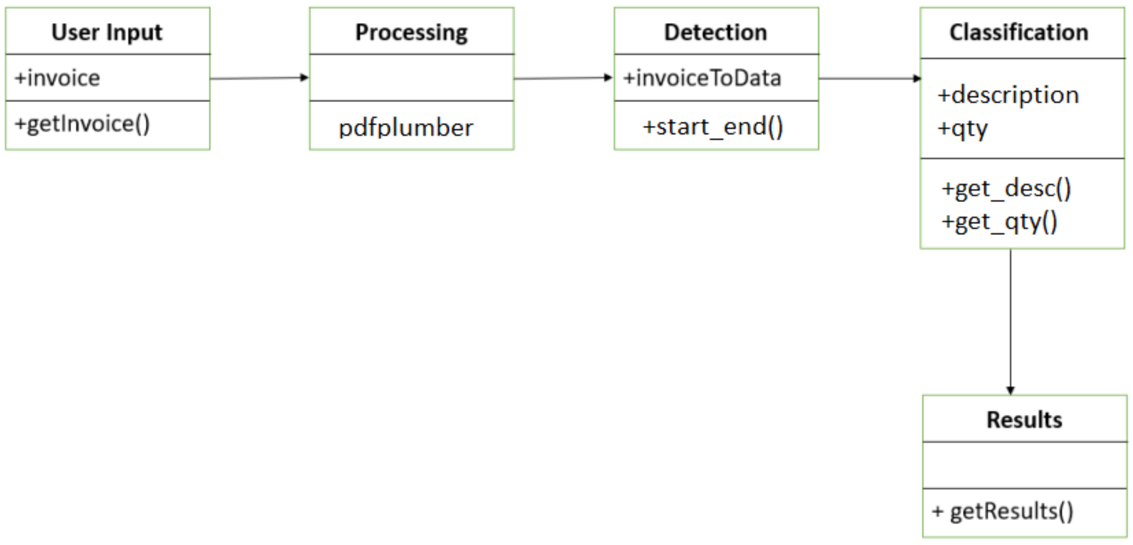


Figure 3.5: Class Diagram of Proposed Model

3.3 Software Requirement Specification

A software requirement specification (SRS) is a document that describes what the software will do and how will it be expected to perform. It also describes the functionality the product needs to fulfil all stakeholders (business, users) needs. Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. User requirements express how a facility, equipment or process should perform in terms of the product to be manufactured, required throughput, and conditions in which product should be made

3.3.1 Functional Requirements:

- **OCR Integration**: The OCR software must be integrated with the invoice processing system to extract data from invoices accurately.
- **OCR Scanning**: The OCR software should be able to scan invoices in various formats including PDF, JPEG, and TIFF.
- **Data Extraction**: The OCR software must be able to extract data from different sections of the invoice such as part, description, total, quantity and price.
- **Data Validation**: The OCR software should validate the extracted data against predefined rules to ensure accuracy and consistency.
- **Data Export**: The OCR software should export the extracted data in a standard format such as XML or CSV to facilitate integration with other systems.

3.3.2 Non functional Requirements:

- **Accuracy**: The OCR software should have a high level of accuracy in recognizing and extracting data from invoices to avoid errors that could impact financial records.
- **Speed**: The OCR software should process invoices quickly to minimize the time between receiving invoices and processing them for payment.
- **Scalability**: The OCR software should be able to handle large volumes of invoices as the business grows without compromising performance.
- **Security**: The OCR software should be secure and comply with data protection regulations to ensure that invoice data is kept confidential and protected against unauthorized access or theft.

- **Reliability**: The OCR software should be reliable and operate with high availability to minimize downtime and ensure that invoices are processed without interruption.

3.3.3 Domain Requirements:

- Invoice processing using OCR (Optical Character Recognition) technology typically involves the following domain requirements:
- **PDF pre-processing**: The invoice image needs to be pre-processed to ensure that it is of good quality and is suitable for OCR analysis. This may involve tasks such as de-skewing, noise reduction, contrast enhancement, and image normalization.
- **Text recognition**: OCR technology is used to recognize the text on the invoice, which can then be extracted and used to populate fields in a database or accounting system. This process may involve recognizing different types of characters, such as numbers, letters, and symbols.
- **Data extraction**: Once the text has been recognized, the relevant data needs to be extracted from the invoice. This may involve identifying and extracting information such as part, description, total, quantity and price.

CHAPTER 4

PROJECT PLANNING

4.1 Project Planning and Scheduling

As shown in the below figure, different phases involved in the project are,

Problem analysis: In this phase, we decided upon the problem to be solved for the project/ decide upon the topic for the project. It is followed by the Planning phase: In this phase, we laid out a plan to complete various activities related to the project. The next phase is the Project literature survey: In this phase, we have surveyed different types of technical papers related to the problem which we have picked for our project. The next phase is Study of DL concepts, where we tried to learn the deep learning concepts required for the project. In the next phase, that is, Collection of datasets, we collected the invoices. The next phase is the System Design and architecture where we design the architecture of the proposed system. After design phase, next phase is the Implementation phase, where we implement our proposed system using Python. After the implementation is completed, the next phase is to Test the system. The last phase is preparation of the final report, in this phase we provide a final and detailed report of our project.

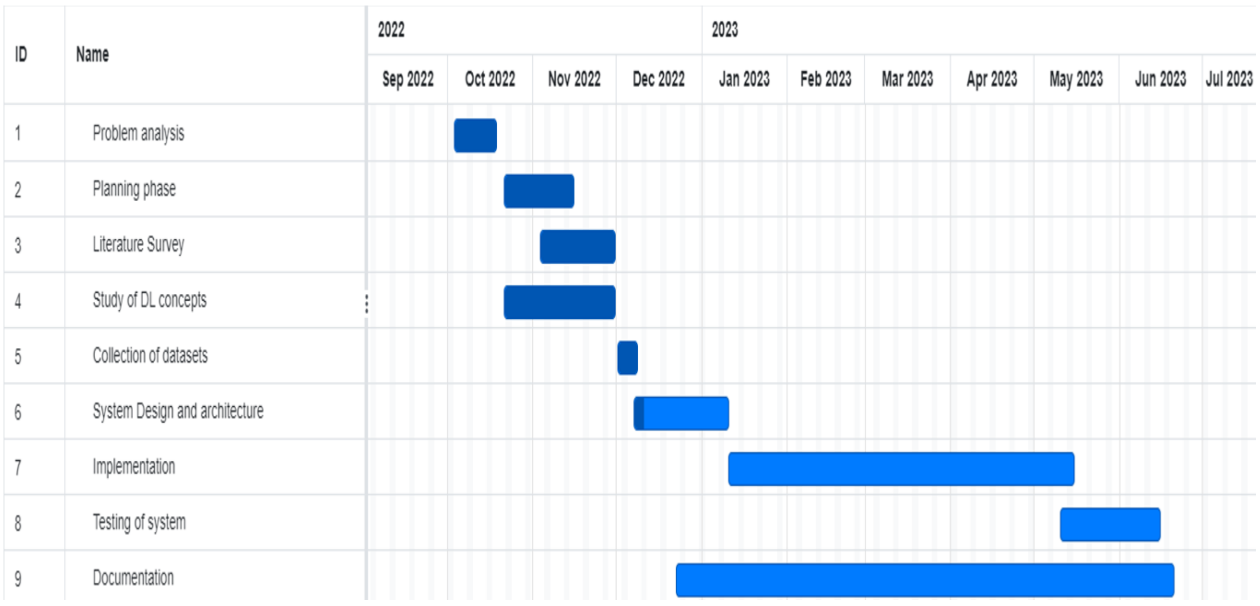


Figure 4.1.1: Project Planning

CHAPTER 5

SYSTEM DESIGN

5.1 System Architecture

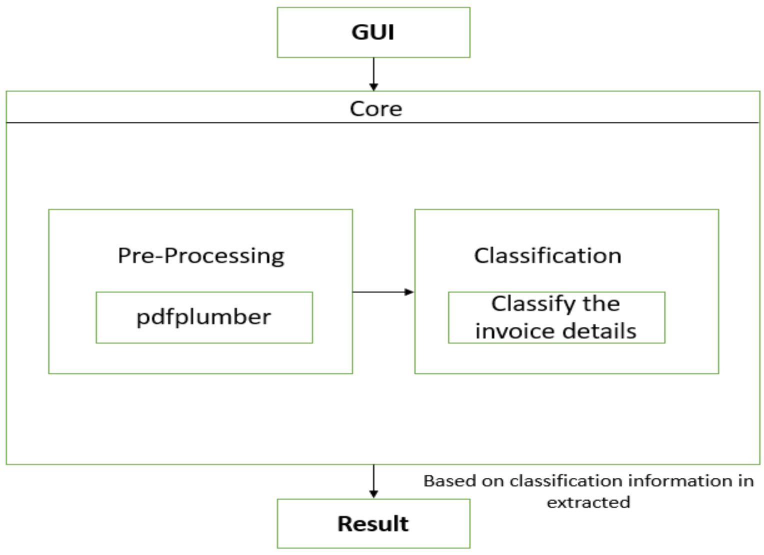

GUI
Core
Pre-Processing
pdfplumber
Classification
Classify the invoice details
Based on classification information in extracted
Result

Figure 5.1.1 System Architecture

5.2 Component Design/Module Decomposition

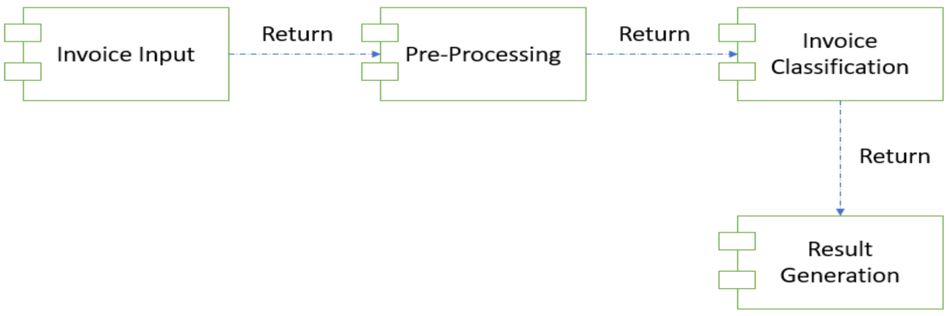


Figure 5.2.1 Component Design

Module Decomposition

Invoice Input:

- The user must have datasets of invoices that are to be processed.
- The invoices provided by the users must be of same format.
- The user can then use the application’s interface to select the PDF for classification of the invoice data.

Pre-Processing

- Based on invoice input provided by the user the next immediate step is to pre-process the invoice.
- Because it can help to standardize the invoice in the provided dataset.
- This module helps to differentiate all the other unwanted data in the invoice and to process with the required one.

Invoice Classification

- After differentiating the relevant areas in the provided invoice, the next step is to classify them.
- The pdfplumber library helps to classify/extract the relevant data from invoice.

Result Generation

- Once the invoice is processed, the processed data is sent to excel sheet.
- The path where the excel is also given by the user to store the processed data.

5.3 Interface Design

The process of defining the interfaces between system components is known as interface design. This interface description must be clear. A component can be used without other components needing to know how it is implemented if it has a clear interface. The components can be planned and developed simultaneously once interface criteria have been agreed upon. The front-end application view with which a user interacts in order to use the software is known as the user interface.

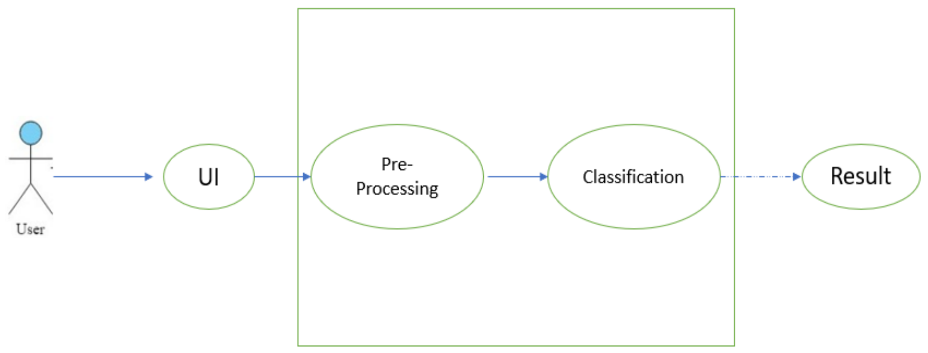


Figure 5.3.1: Interface Design for invoice processing

5.4 Data Structure Design

Array: Array is a data structure consisting of a collection of elements, each identified by at least one array index or key. Array is used to store the predicted values of trained model to evaluate performance.

List: A list is a collection of items that can be of different data types, such as integers, strings, or even other lists. Lists are ordered and mutable, which means that their contents can be changed.

Dictionary: A dictionary is a general-purpose data structure for storing a group of objects. A dictionary is an ordered or unordered list of key-element pairs, where keys are used to locate elements in the list.

Tuple: Tuples are used to store multiple items in a single variable and it can have any number of items that they may be of different types (integer, float, list, string, etc.). A tuple can also be created without using parentheses.

5.5 Algorithm Design

1. Optical Character Recognition (OCR) - These algorithms are used to identify and extract text from the images of scanned PDF documents.
2. Data extraction algorithms - These algorithms are used to extract data from tables, forms, and other structured elements within a PDF document.

Pdfplumber uses a combination of OCR and Data Extraction algorithms to identify and extract text and data from PDF documents. It can be used for a variety of tasks, such as extracting tables or forms from financial documents, extracting text from legal contracts or extracting data from invoices.

Algorithm - OCR

1. Load the input pdf

2. Pre-process the pdf to improve its quality

3. Identify regions in the pdf that are likely to contain text (text detection)

4. For each text region:

    *Segment individual characters (character segmentation)

    *Extract features that describe the shape, texture, and other characteristics of each character (feature extraction)

    *Compare the extracted features to a database of known characters to determine the most likely character class (classification)

    *Apply post-processing techniques to correct errors and improve accuracy

5. Concatenate the recognized characters to form words and sentences

6. Output the recognized text

Algorithm – Data Extraction

1. Load the input image or document
2. Perform layout analysis to identify the location of the data to be extracted, such as tables or fields in a form
3. Identify the boundaries of the data to be extracted, such as the rows and columns of a table or the fields in a form
4. For each field or cell to be extracted:
    - Perform OCR to recognize the text in the field or cell
    - Apply post-processing techniques to correct errors and improve accuracy

  - Convert the recognized text to a structured format, such as a database record or spreadsheet cell

5. Apply additional post-processing techniques to improve accuracy and completeness, such as verifying extracted data against a known database or correcting errors in the extracted data

Output the final extracted data in the desired format, such as a CSV file or database table.

CHAPTER 6

IMPLEMENTATION

6.1 Implementation Approaches

Define the project's scope to define which precise data elements, such as the vendor name, invoice number, and line items, you want to extract from the PDFs of the invoices and how you want to save that data in the Excel file.

Install and setup the essential tools, including pdfPlumber and any additional dependencies for your project. To ensure that your PDFs are properly converted to machine-readable format, you might additionally need to configure an OCR engine.

Create your Python programme: To extract data from invoices and store it in Excel, create a Python script. Use the csv module of pdfPlumber to write the relevant data to an Excel file after parsing the required information from the PDFs.

Test and improve the script: To guarantee that the script can handle various layouts and formatting styles, test it using a range of invoice PDFs. Adjust the script as needed.

Script deployment: Implement the script in a live environment. Create a batch processing system to execute the script on fresh PDF invoices as they come in.

Watch for problems and maintain the script: Keep an eye out for script maintenance needs while you process invoices. Periodically check the extracted data for accuracy, and update the script as necessary to address any new invoice formats or data needs that appear over time.

6.2 Coding Details

6.2.1 Core Module

The core module where keywords like part, description, quantity, price and amount is targeted from the pdf , details relevant to those keywords are processed and then extracted and stored in a separate excel file.

```
def process_pdf(self):
    lines=[]
    pdf = pdfplumber.open(self.pdf_path)
```

```
#go through all the pages
for page in pdf.pages:
    #go through all lines
    for line in page.extract_text().splitlines():
        #replaces one or more whitespaces with single whitespace
        lines.append(re.sub('\s+', ' ', line.strip()))

start, end = self.start_end(lines)

if start == [] or end == []:
    print('keyword not found')
    return

else:
    print("keyword found")
    start, end = min(start), max(end)

    #dictionary that contains the result
    result = {
        'PART':[],
        'DESCRIPTION':[],
        'QTY':[],
        'PRICE':[],
        'TOTAL':[],
    }

    for data in lines[start:end]:
        if 'part' not in data.lower() and 'description' not in data.lower():
            result['PART'].append(data.split()[0])
            result['TOTAL'].append(data.split()[-1])
        result['PRICE'].append(data.split()[-2])
        result['QTY'].append(data.split()[-3])
        result['DESCRIPTION'].append(' '.join(data.split()[1:-3]))

    df = pd.DataFrame(result)
    excel_path = self.pdf_path.replace(".pdf", ".xlsx")
    df.to_excel(excel_path, index=False)

    print("Result saved in", excel_path)
```

CHAPTER 7

TESTING

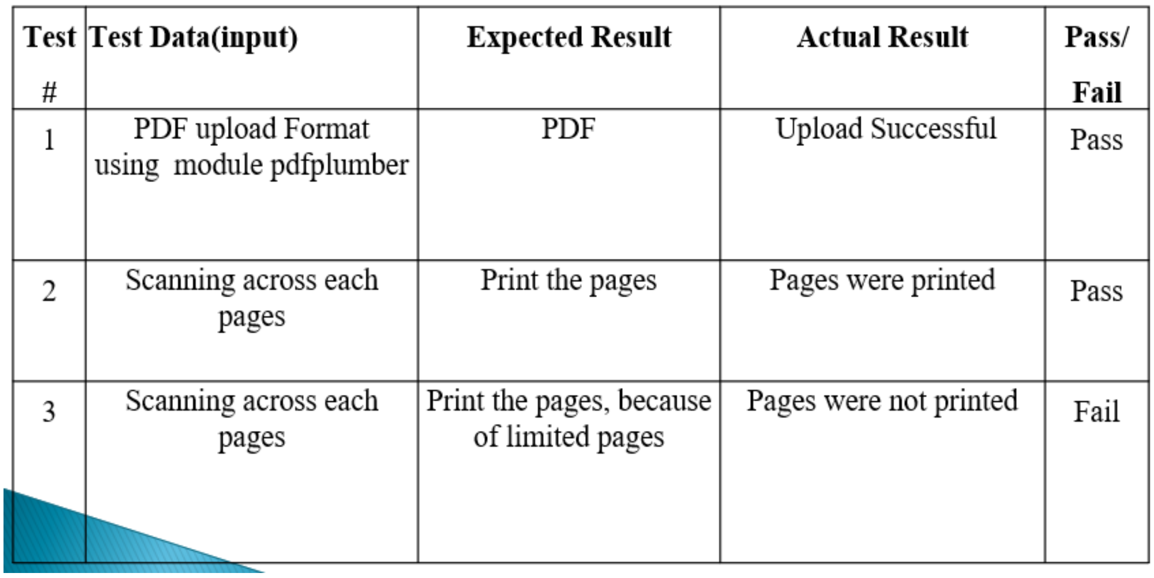
7.1 Testing Approach

7.1.1 Unit Testing:

Unit testing is a testing method in which the developer personally tests individual modules to ascertain whether there are any bugs. It is concerned with the standalone modules' functionality. Developers perform unit testing while creating an application (the coding phase).

Unit testing is to separate out each component of the program and demonstrate that each component is accurate. Early in the development cycle, problems are discovered by unit testing. This covers errors in the unit's definition as well as issues in the programmer's implementation. Typically automated, unit tests can be run frequently throughout the development process. Write, run, and manage unit tests more quickly by using testing frameworks and tools like JUnit for Java or pytest for Python, according to developers. These frameworks offer practical tools for test setup, organization, and result reporting.

Table 7.1: Unit Testing Detection



Test #	Test Data(input)	Expected Result	Actual Result	Pass/ Fail
1	PDF upload Format using module pdfplumber	PDF	Upload Successful	Pass
2	Scanning across each pages	Print the pages	Pages were printed	Pass
3	Scanning across each pages	Print the pages, because of limited pages	Pages were not printed	Fail

Table 7.2 : Unit Testing Classification

Test #	Test Data(input)	Expected Result	Actual Result	Pass/ Fail
4	Target the respective keywords in the lines	Keywords like, PART, DESCRIPTION,QTY, PRICE and TOTAL	Keywords were not targeted because of wrong keyword	Fail
5	Store the content of discovered keywords in the Lists	Gather all the content into the lists.	Contents are dumped to the lists as expected.	Pass
6	Display of information in console and store in an excel file	Display and generate a new excel file.	Information were displayed and also stored in a separate excel file.	Pass

7.1.2 Integrated Manual Testing

Integration tests ensure that the various components or services used by your application function properly. An application's ability to send and process requests as it should be is tested by manual testers who mimic every stage of the user journey and make sure that users receive only the information they need to view. Software testing of this kind, known as integrated manual testing, is concerned with examining how several parts or modules of a system work together and independently of one another. It entails testing the integrated system as a whole to make sure that all functions properly.

Table 7.3: Integrated Manual Testing

Test #	Test Data(input)	Expected Result	Actual Result	Pass/ Fail
1	Provide PDF to process to the system without pdfplumber installed	Data Extraction and excel file generation	pdfplumber library is missing	Fail
2	Data set is given to the system with pdfplumber is installed	Data Extraction and excel file generation	Data extracted successfully	Pass
3	Use Pandas to store data from console to excel	Data Extraction in excel.	Data were stored successfully in excel	Pass

CHAPTER 8

RESULT DESUSSION AND PERFORMANCE ANALYSIS

8.1 Test Reports

Accuracy of Processing of PDF, measured and plotted using matplotlib

Accuracy – 99-100%

Processing Time – 60%

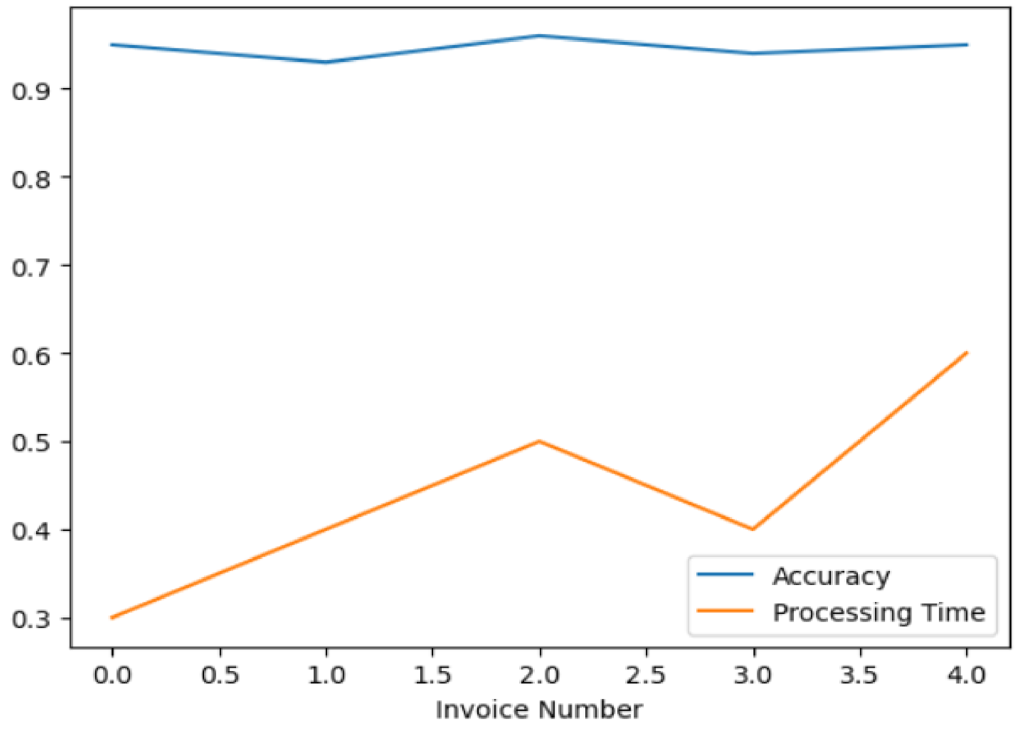


Figure 8.1.1: Result Analysis

The graph shows a comparison between the invoice numbers extracted by the OCR system and the percentage accuracy of those extractions. The system achieved an accuracy rate of between 99-100%, indicating that the OCR system was able to accurately extract the invoice numbers from the scanned invoices.

The graph also shows that the efficiency of the OCR system was 60%, which means that it was able to extract the invoice numbers from 60% of the scanned invoices processed. While this is not a perfect efficiency rate, it indicates that the OCR system was still able to process a significant number of invoices accurately and efficiently.

8.2 Snapshots

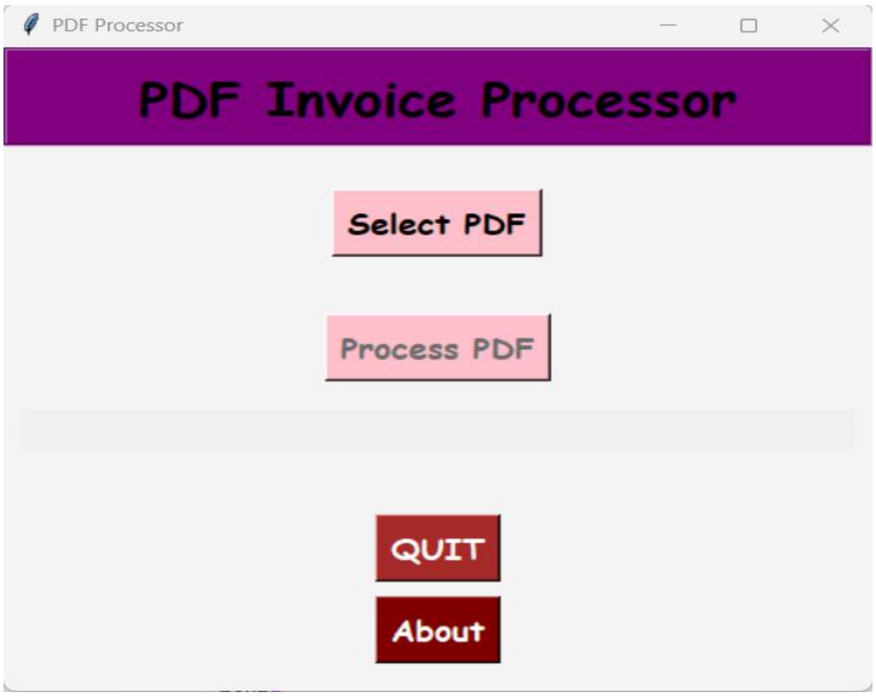
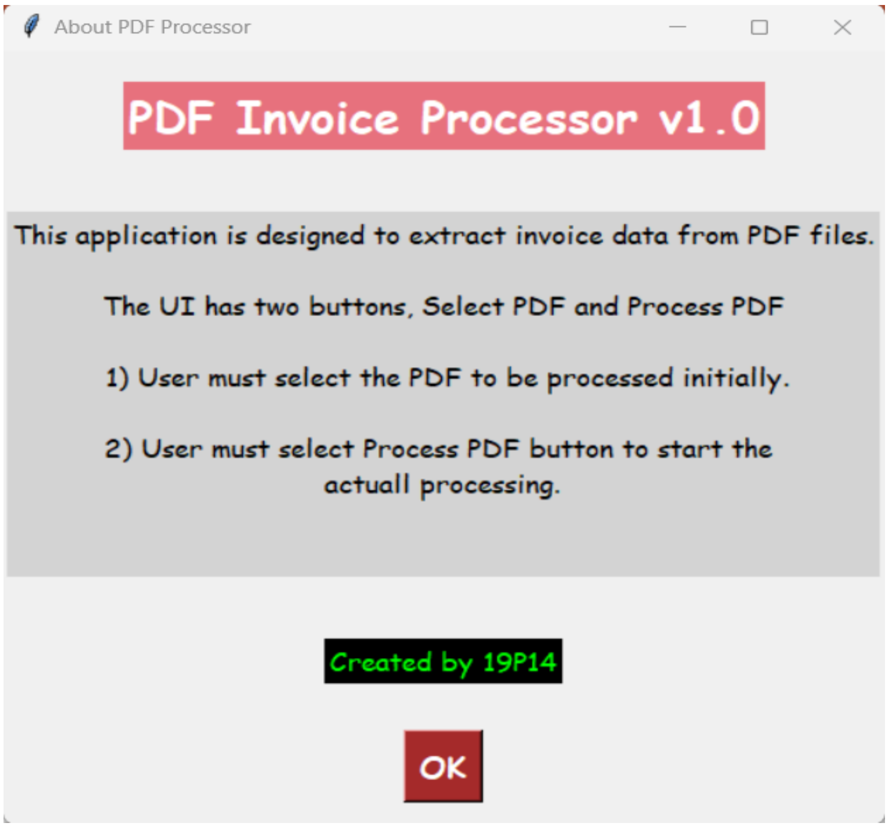


Figure 8.2.1: Home page of PDF Processing



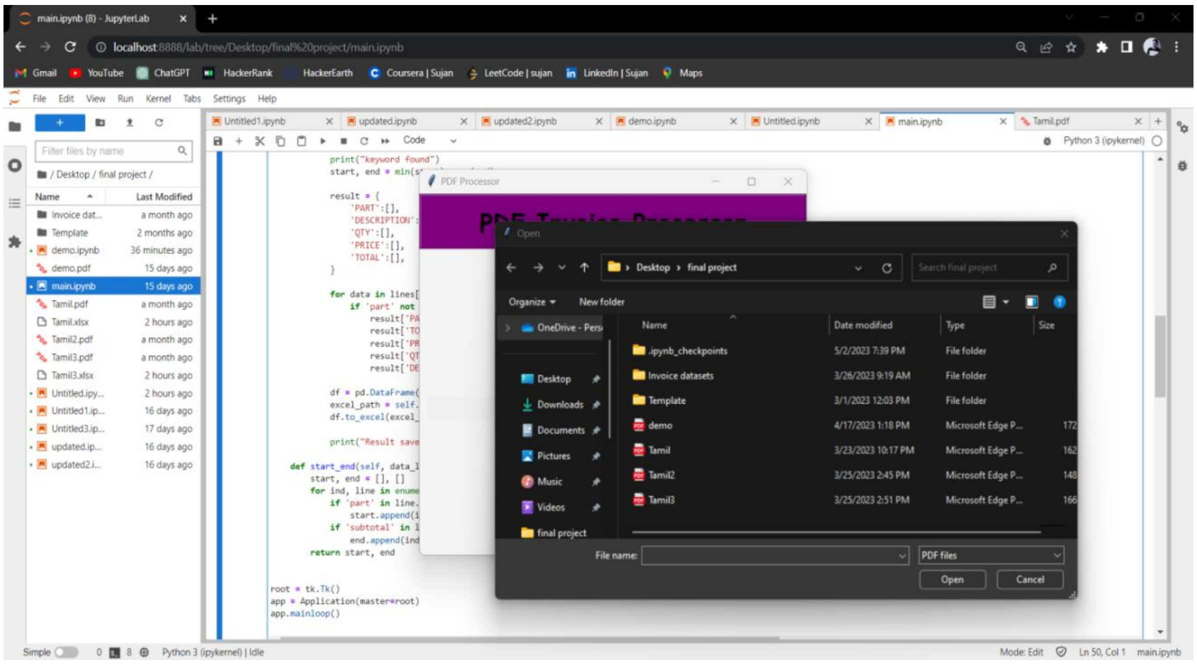
This application is designed to extract invoice data from PDF files.

The UI has two buttons, Select PDF and Process PDF

1) User must select the PDF to be processed initially.

2) User must select Process PDF button to start the actuall processing.

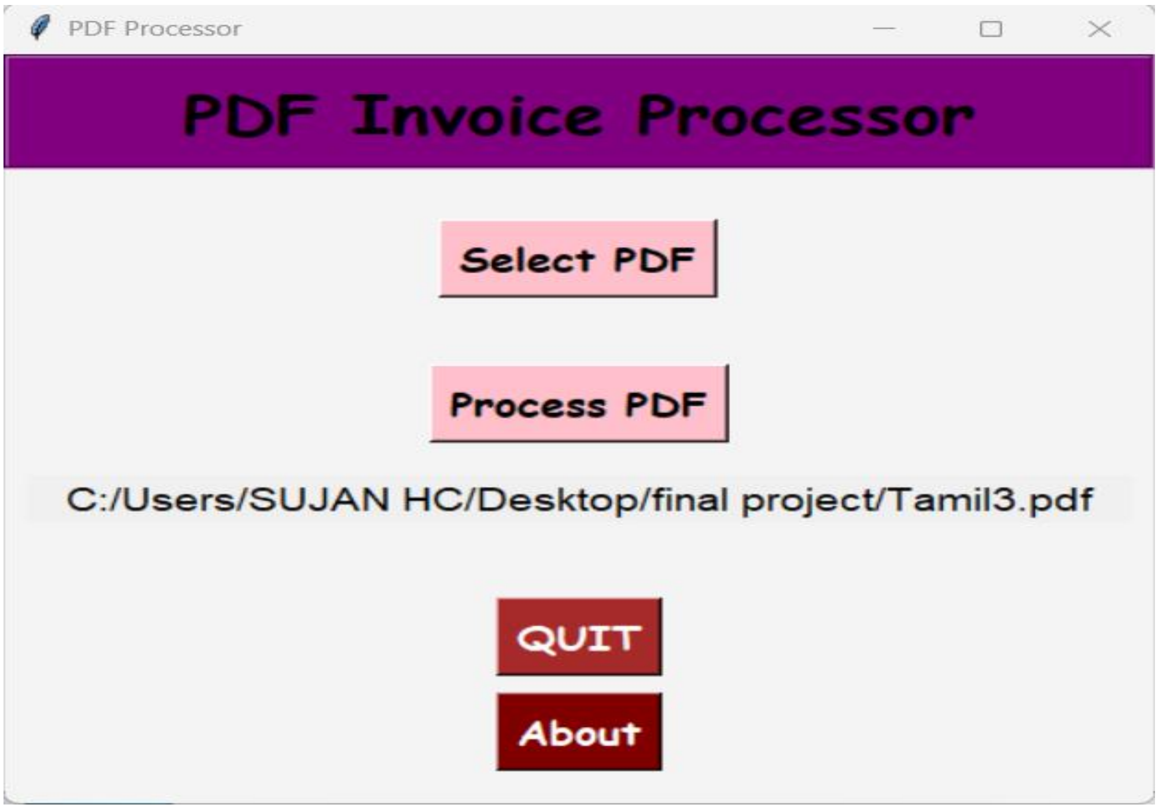
Figure 8.2.2: About page of PDF Processing



```
                print("keyword found")
                start, end = min(s

                result = {
                    'PART':[],
                    'DESCRIPTION':
                    'QTY':[],
                    'PRICE':[],
                    'TOTAL':[],
                }

                for data in lines[
                    if 'part' not
                        result['PA
                        result['TO
                        result['PR
                        result['QT
                        result['DE

                df = pd.DataFrame(
                excel_path = self.
                df.to_excel(excel_

                print("Result save

        def start_end(self, data_l
            start, end = [], []
            for ind, line in enume
                if 'part' in line.
                    start.append(i
                if 'subtotal' in l
                    end.append(ind
            return start, end


root = tk.Tk()
app = Application(master=root)
app.mainloop()
```

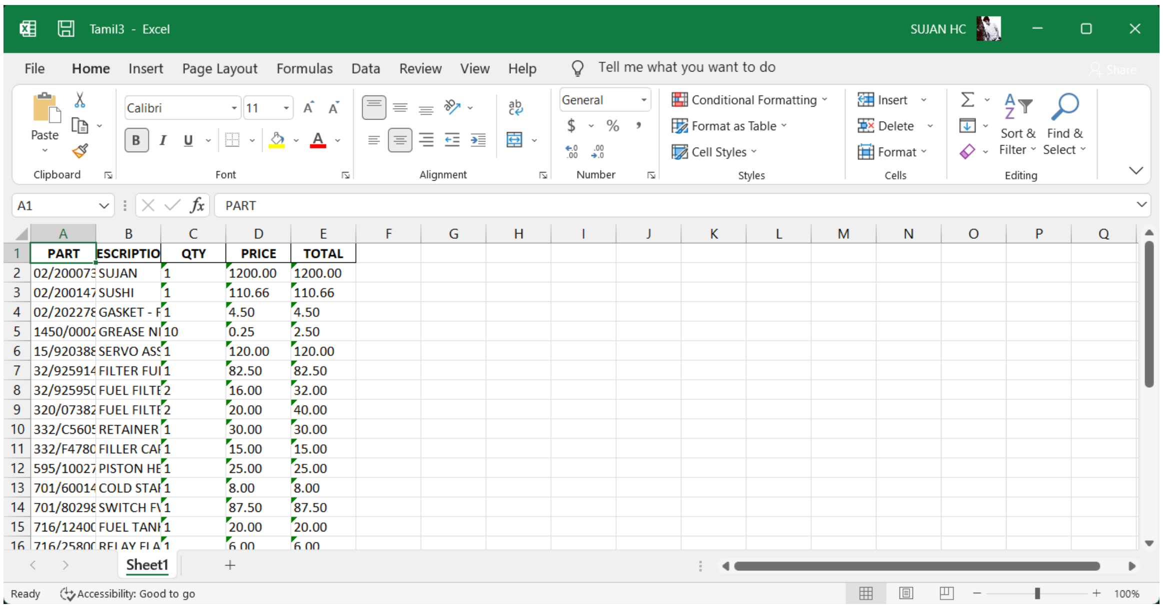
Name	Date modified	Type	Size
.ipynb_checkpoints	5/2/2023 7:39 PM	File folder	
Invoice datasets	3/26/2023 9:19 AM	File folder	
Template	3/1/2023 12:03 PM	File folder	
demo	4/17/2023 1:18 PM	Microsoft Edge P...	172
Tamil	3/23/2023 10:17 PM	Microsoft Edge P...	162
Tamil2	3/25/2023 2:45 PM	Microsoft Edge P...	148
Tamil3	3/25/2023 2:51 PM	Microsoft Edge P...	166

Figure 8.2.3: Users are asked to select pdf



C:/Users/SUJAN HC/Desktop/final project/Tamil3.pdf

Figure 8.2.4: UI after selecting the PDF



PART	ESCRIPTIO	QTY	PRICE	TOTAL
02/200073	SUJAN	1	1200.00	1200.00
02/200147	SUSHI	1	110.66	110.66
02/202278	GASKET - F	1	4.50	4.50
1450/0002	GREASE NI	10	0.25	2.50
15/920388	SERVO ASS	1	120.00	120.00
32/925914	FILTER FUI	1	82.50	82.50
32/925950	FUEL FILTE	2	16.00	32.00
320/07382	FUEL FILTE	2	20.00	40.00
332/C5605	RETAINER	1	30.00	30.00
332/F4780	FILLER CAI	1	15.00	15.00
595/10027	PISTON HE	1	25.00	25.00
701/60014	COLD STAI	1	8.00	8.00
701/80298	SWITCH FV	1	87.50	87.50
716/12400	FUEL TANI	1	20.00	20.00
716/25800	RELAY FLA	1	6.00	6.00

Figure 8.2.5: Result is stored in excel

CHAPTER 9

CONCLUSIONS, APPLICATIONS AND FUTURE WORK

9.1 Conclusion

Processing invoices is a crucial company activity that involves handling and managing financial records. To preserve financial stability and guarantee on-time payments, firms must process invoices precisely and quickly. Processing invoices is now much more accurate and efficient thanks to technology like OCR.

9.2 Applications

- **Invoice management software**: Invoice management software can automate the entire invoice processing workflow, from receipt of the invoice to payment. It can also provide real-time visibility into invoice status, track approvals, and manage exceptions.
- **Invoice data validation**: OCR technology can help to validate the accuracy of invoice data by comparing the extracted data with pre-defined criteria. This ensures that the invoice data is correct and complies with the company's policies and procedures
- **Fraud detection**: OCR technology can be used to identify potential instances of fraud or duplicate invoices. This can help to reduce the risk of financial loss and improve overall compliance.
- **Data extraction and verification**: OCR technology can be used to extract data from invoices and verify the accuracy of the data against other sources, such as purchase orders or delivery notes.

9.3 Future Scope of Work

Some future enhancements about invoice processing using OCR

- Improved accuracy through the use of machine learning algorithms to enhance recognition accuracy and reduce errors associated with manual data entry.
- Integration with other technologies such as AI and natural language processing to enhance the efficiency and accuracy of invoice processing.

- Improved handling of unstructured data to enable OCR technology to process invoices that are not standardized.
- Adoption of AI-based solutions to provide real-time data analytics and insights, enabling businesses to make better decisions based on invoice data

REFERENCES

[1] Deepali Baviskar Swathi Ahirrao,Ketan Kotacha **“*Dataset for template free invoice processing and its evaluation using AI approaches*”** vol. 9, Feb 2021

[2] David A Kosiba,Rangachar Kasturi **“*Invoice structure analysis*”**, Jul 2021.

[3] Harshith Sidhwa,Sahil Malhotra **“*Extraction of information from bill receipts using optical character recognition*”**, Oct 2020.

[4] Najoua Rahal, Maraua Taunsiet al.**, “*Information Extraction From Arabic and Latin Scanned Invoices*”**, Aug 2021.

[5] Danish Sholanki ,Dilip **“*Method of Robotic Process Automation invoice processing and mailing*”,** Aug 2022.

[6] Trapti Upadaya Shashank Chauhan,Ujwal sinha **“*Invoice Processing Automation*”,** Apr 2021.

[7] Xufung Ming gao, Dong wang, **“*Intelligent Document Processing based on RPA and machine learning*”,** Jan 2020.

[8]Bharath V and N Shobha Rani, **“*A Font style classification system for English OCR*”,** Jan 2017

[9] Rohit Saluja, Devaraj Adiga et al.**, “*Error Detection and Corrections in Indic OCR using LSTMs*”**, 2017.

[10] Harshit Sidhwa, Sudhanshu Kulshrestha, Sahil Malhotra, Shivani Virmani **“*Text Extraction from Bills and Invoices”***, May 2018.