Defending Against Poisoning Attacks in Federated  
Learning With Blockchain

## A MAIN PROJECT REPORT

***Submitted by***

|  |  |
| --- | --- |
| **GOKUL P** | **(622221104017)** |
| **SURYA S** | **(622221104057)** |
| **THIRUMALAI VASAN L** | **(622221104060)** |
| **VIGNESH P G** | **(622221104062)** |

***In partial fulfilment for the award of the degree of***

## BACHELOR OF ENGINEERING

***in***

**COMPUTER SCIENCE AND ENGINEERING**

# PAAVAI COLLEGE OF TECHNOLOGY, NAMAKKAL.

**MAY - 2025**

# BONAFIDE CERTIFICATE

Certified that this project report **“DEFENDING AGAINST POISONING ATTACKS IN FEDERATED LEARNING WITH BLOCKCHAIN”** is the bona fide word of **GOKUL P (622221104017), SURYA S (622121104057), THIRUMALAI VASAN L (622221104060), VIGNESH P G (622221104062)** who have carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE** | **SIGNATURE** |
| **Mr. M.SUNDARAM, M.E.,** | **Mr. M.SUNDARAM , M.E.,** |
| **HEAD OF THE DEPARTMENT** | **SUPERVISOR** |
| Department of Computer Science and Engineering | Department of Computer Science and Engineering |
| Paavai College of Technology, | Paavai College of Technology, |
| Namakkal-637018 | Namakkal-637018 |

Submitted for the university project viva-voce held on………………..........

|  |  |
| --- | --- |
| **INTERNAL EXAMINER** | **EXTERNAL EXAMINER** |

# DECLARATION

We **GOKUL P, SURYA S, THIRUMALAI VASAN L** and **VIGNESH P G** hereby declare that the project report titled done by us under the guidance of **Mr. E. ELANCHEZHIYAN, M.E.,** at **PAAVAI COLLEGE OF TECHNOLOGY, NAMAKKAL** is submitted in partial fulfilment of the requirements for the award of **BACHELOR OF ENGINEERING** degree in **COMPUTER SCIENCE AND ENGINEERING**. Certified further that, to the best of my knowledge, the work reported here in does not form part of any other project report or dissertation on the basic of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**1.**

**2.**

**3.**

**4.**

**DATE:**

**PLACE: NAMAKKAL SIGNATURE OF THE CANDIDATES**

# ACKNOWLEDGEMENTS

A great deal of arduous work and effort has been spent in implementing this project work. Several special people have guided us and have contributed significantly to this work and so this becomes obligatory to record our thanks to them.

We express our profound gratitude to our beloved chairman **Shri. CA.N.V.NATARAJAN, B.Com, F.C.A.,** and our correspondent **SMT.N.MANGAINATARAJAN, M.Sc.,** for giving motivation and providing all necessary facility for the successful completion of my project.

We wish to express my sincere thanks to our respected Director Administration, **Dr.K.K.RAMASAMY, M.E., Ph.D.,** for all the blessing and help provided during required time to complete the same.

We would like to thank out respected Principal **Dr.M.PREM KUMAR, M.E., Ph.D.,** for their moral support and deeds in bringing out this project successfully.

We extend our gratefulness to **MR. M.SUNDARAM, M.E.,** Head of the Department**,** for the able guidance and useful suggestions, which helped us for completing project work in time.

We express our sincere thanks to **MR. M.SUNDARAM, M.E.,** project Coordinator for the useful suggestions, which helped us for completing the project in time.

We would like to extend my sincere thanks to **MR. M.SUNDARAM, M.E.,** Supervisor, for giving me an opportunity to do this project and also for his inspiring guidance, generous help and support.

We would like to extend my sincere thanks to **all our department staff members, and our parents** for their encouragement and advice to do the project work with full interest and enthusiasm.

# ABSTRACT

Federated Learning (FL) has emerged as a transformative approach in machine learning, enabling decentralized, privacy-preserving data processing across distributed clients. By allowing multiple clients to collaboratively train a global model without exposing sensitive local data, FL addresses key challenges in privacy, scalability, and data security. Despite its numerous benefits, FL remains vulnerable to various types of data poisoning attacks, among which Label-Flipping Attacks (LFA) pose a significant threat. In Label-Flipping Attacks, malicious clients deliberately alter the labels of local datasets, leading to erroneous updates in the global model, which can misclassify data and compromise the model’s integrity and accuracy. To counter these challenges, this project proposes a novel approach combining blockchain technology with a Self-Purified FL (SPFL) model verification system. In this blockchain-integrated federated learning framework, global model labels are securely stored in a decentralized blockchain, ensuring immutability and transparency. Whenever a client intends to train the model with their local data, the blockchain system activates the SPFL verification process to assess the validity of the data. This verification step ensures that only legitimate, non-malicious data contributes to the global model training process. Upon receiving a request from a client to participate in model training, the blockchain prompts the SPFL mechanism to assess the accuracy of the local data labels. If any malicious behavior, such as Label-Flipping Attacks, is detected, the client is eliminated from the training process. Only after successful verification of the local data, where no malicious activity is identified, can the client’s data be used to update and refine the global model. This ensures that the model maintains high accuracy and integrity, even in the presence of malicious clients. The integration of blockchain technology guarantees data security and traceability, while the SPFL mechanism eliminates the threat of mislabelled data, safeguarding the global model from corruption.

|  |  |  |
| --- | --- | --- |
|  | **TABLE OF CONTENTS** |  |
| CHAPTER | **TITLE** | PAGE NO |
| - | **ABSTRACT** | V |
| - | **LIST OF FIGURES** | IX |
| - | **LIST OF ABBREVIATIONS** | X |
| 1 | **INTRODUCTION** | 1 |
| 1.1 | **Project Introduction** | 1 |
| 1.2 | **Objectives** | 2 |
| 2 | **LITERATURE SURVEY** | 3 |
| 3 | **SYSTEM ANALYSIS** | 8 |
| 3.1 | **Existing System** | 8 |
| 3.1.1 | **Disadvantages** | 8 |
| 3.2 | **Proposed System** | 9 |
| 3.2.1 | **Advantages** | 9 |
| 3.3 | **Feasibility Study** | 10 |
| 3.3.1 | **Technical Feasibility** | 10 |
| 3.3.2 | **Operational Feasibility** | 11 |
| 3.3.3 | **Financial Feasibility** | 12 |
| 4 | **PROJECT DESCRIPTION** | 13 |
| 4.1 | **Problem Definition** | 13 |
| 4.2 | **Overview of the Project** | 13 |
| 4.2.1 | **Core Components** | 13 |
| 4.2.2 | **Existing Challenges Addressed** | 14 |
| 4.2.3 | **Modules and Architecture** | 14 |
| 4.2.4 | **Software and Tools Used** | 15 |
| 4.2.5 | **Future Enhancement** | 15 |
| 5 | **MODULE DESCRIPTION** | 16 |
| 5.1 | **Audio Input and Preprocessing Module** | 16 |
| 5.2 | **Text Processing and Mapping** | 16 |
| 5.3 | **Sign Visualization Module** | 16 |
| 5.4 | **Controller and GUI Module** | 16 |
| 6 | **REQUIREMENTS ENGINEERING** | 17 |
| 6.1 | **Software Requirements** | 17 |
| 6.2 | **Functional Requirements** | 17 |
| 7 | **SOFTWARE DESCRIPTION** | 18 |
| 7.1 | **Programming Language: Python 3.7.4** | 18 |
| 7.2 | **Libraries and Tools Used** | 18 |
| 7.2.1 | **SpeechRecognition** | 18 |
| 7.2.2 | **Numpy** | 18 |
| 7.2.3 | **Matplotlib** | 19 |
| 7.2.4 | **OpenCV** | 19 |
| 7.2.5 | **Tkinter and EasyGUI** | 19 |
| 8 | **SYSTEM DESIGN** | 21 |
| 8.1 | **System Architecture** | 21 |
| 8.2 | **Data Flow Diagram** | 22 |
| 8.2.1 | **Level 0** | 22 |
| 8.2.2 | **Level 1** | 23 |
| 8.3 | **Block Diagram** | 24 |
| 9 | **CONCLUSION AND FUTURE ENHANCEMENT** | 26 |
| 9.1 | **Conclusion** | 26 |
| 9.2 | **Future Enhancement** | 26 |
| 10 | **APPENDIX** | 28 |
| 10.1 | **Source Code** | 28 |
| 10.2 | **Output Screenshots** | 33 |
| 11 | **REFERENCES** | 35 |

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **TITLE** | **PAGE NO** |
| 8.1 | DOCUMENT DIGITIZATION ARCHITECTURE | 21 |
| 8.2.1 | DATA FLOW DIAGRAM LEVEL 0 | 23 |
| 8.2.2 | DATA FLOW DIAGRAM LEVEL 1 | 24 |
| 8.3 | DOCUMENT DIGITIZATION BLOCK DIAGRAM | 25 |

**LIST OF ABBREVATION**

AI - ARTIFICIAL INTELLIGENCE

OCR - OPTICAL CHARACTER RECOGNITION CV - COMPUTER VISION

NLP - NATURAL LANGUAGE PROCESSING ML - MACHINE LEARNING

HTR - HANDWRITTEN TEXT RECOGNITION JSON - JAVASCRIPT OBJECT NOTATION

DOCX - MICROSOFT WORD DOCUMENT FORMAT AWS - AMAZON WEB SERVICES

REST - REPRESENTATIONAL STATE TRANSFER API - APPLICATION PROGRAMMING INTERFACE NER - NAMED ENTITY RECOGNITION

JWT - JSON WEB TOKEN

POS - PART OF SPEECH

BER - BIDIRECTIONAL ENCODER REPRESENTATION NLTK - NATURAL LANGUAGE TOOLKIT

CLI - COMMAND LINE INTERFACE

# CHAPTER 1 INTRODUCTION

## PROJECT INTRODUCTION

In today's fast-paced digital world, the need to convert handwritten documents into structured, editable formats is more pressing than ever. Historical manuscripts, academic notes, legal records, and various handwritten documents hold valuable information, but their accessibility remains limited due to the challenges of digitization. Traditional OCR systems often struggle with handwritten text, varying writing styles, and complex document layouts. This project introduces an AI-powered handwritten document digitization system that leverages deep learning- based OCR models, computer vision, and natural language processing (NLP) to overcome these limitations. The system accurately extracts handwritten text, digitizes hand-drawn diagrams, and organizes structured content into high- quality Word documents with minimal data loss. By incorporating state-of-the-art AI techniques, it ensures high accuracy, multilingual support, and document layout preservation, making it a robust solution for diverse applications. Deep learning-based OCR models such as Tesseract and TrOCR enable high-accuracy extraction of handwritten content from scanned documents or images. These models utilize convolutional neural networks (CNNs) and transformer-based architectures to enhance text recognition across varying handwriting styles, ensuring optimal readability and conversion precision. The system also supports multilingual handwriting recognition, allowing seamless processing of documents in multiple languages. Computer vision techniques, including CNNs and OpenV, are used to detect, classify, and digitize hand-drawn shapes and diagrams. The system accurately preserves the spatial structure

and relationships between different elements, converting sketches into vectorized formats that integrate smoothly into digital documents. This ensures that diagrams retain their intended meaning and clarity when transferred to a structured document format.

## OBJECTIVE

The primary objective of this project is to develop an AI-powered system that accurately converts handwritten documents into structured, digital formats while preserving their integrity. The system focuses on leveraging deep learning-based OCR models to enhance the accuracy of handwritten text recognition across various handwriting styles and languages. Additionally, it integrates computer vision techniques to detect, classify, and digitize hand- drawn diagrams and sketches, ensuring that spatial structure and graphical elements are maintained. Ensuring document layout integrity is a key aspect of this project, as it preserves text placement, formatting, and multilingual handwriting recognition for high-quality digital output. Furthermore, NLP- driven automation is employed to organize extracted text into well-structured Word documents, ensuring proper headings, paragraphs, bullet points, and tables. Ultimately, this system aims to improve the accessibility and usability of handwritten records across various domains, including historical preservation, academia, and legal documentation.

## CHAPTER 2 LITERATURE SURVEY

* + 1. **TITLE:** A Survey of Optical Character Recognition Techniques

**AUTHOR:** R. Smith

## YEAR OF PUBLICATIONS: 2023

This paper presents a detailed and broad overview of the various techniques used in Optical Character Recognition (OCR) systems. The author divides the study into multiple stages of the OCR pipeline, including preprocessing, segmentation, feature extraction, and classification. Preprocessing methods such as binarization, noise removal, skew correction, and normalization are discussed as essential steps to prepare the input image. Segmentation focuses on line, word, and character isolation, which is often one of the most difficult parts of OCR, especially with cursive or degraded text. Feature extraction is described as the process of transforming image data into a format suitable for recognition, using methods such as zoning, projections, profiles, and contour following. For classification, the paper explores both traditional and modern techniques: template matching, k-nearest neighbors, decision trees, support vector machines (SVMs), and neural networks. The paper also distinguishes between printed and handwritten OCR, outlining the different challenges and techniques required for each. The study evaluates the performance of several OCR engines available at the time, including Tesseract. The author highlights the need for language models and dictionaries for post-processing, which improve recognition accuracy. Additionally, the paper discusses challenges in multilingual OCR, poor-quality scans, and character overlap.

* + 1. **TITLE:** Optical Character Recognition for Handwritten Historical Documents using LSTM Networks

**AUTHOR:** Alex Graves, Santiago Fernández, Faustino Gomez

## YEAR OF PUBLICATIONS: 2023

This highly influential paper introduces the use of Long Short- Term Memory (LSTM) networks for recognizing handwritten historical documents, which are typically more complex than printed text due to inconsistencies in writing styles, spacing, and ink quality. The authors point out the limitations of traditional OCR methods, especially in cases where segmentation is difficult due to connected characters or cursive handwriting. The key contribution of the paper is the application of Recurrent Neural Networks (RNNs), specifically bidirectional LSTMs, which allow the model to consider both past and future context when making predictions. Unlike traditional methods, this approach does not require explicit segmentation of text into individual characters. Instead, the model learns the sequence of characters directly from the raw image data. The training was conducted using real-world historical manuscripts, and the model achieved remarkable accuracy compared to traditional HMM-based or template-based OCR systems. The paper details the structure of the LSTM network, the training process using Connectionist Temporal Classification (CTC) loss, and the optimization techniques employed. This approach proved to be scalable and adaptable across various languages and scripts, making it a breakthrough in the field of document digitization. The authors also discuss the integration of dictionary-based correction and the potential for real-time OCR applications.

* + 1. **TITLE:** A Comparative Study on OCR Tools for Text Recognition

**AUTHOR:** D. Pratiksha, M. Ranjit

## YEAR OF PUBLICATIONS: 2024

This study presents an in-depth comparative analysis of widely used OCR tools and engines including Tesseract, ABBYY FineReader, and Google Cloud Vision API. The authors set out to evaluate these tools across several parameters such as recognition accuracy, language support, input flexibility, performance on low-quality images, and ease of integration with applications. The experiments involved processing a variety of documents including printed books, scanned PDFs, handwritten notes, and multilingual texts. The authors documented each tool’s performance using standard accuracy metrics (precision, recall, and F-score) as well as usability measures such as ease of use and output formatting capabilities. ABBYY FineReader was found to excel in structured document recognition, especially forms and tables. Google Vision API delivered strong performance in recognizing text from natural scenes and images with mixed content, while Tesseract was recognized for its versatility, customization, and open-source availability. The study highlights how each OCR engine performs differently under various conditions. Tesseract performed best with high- resolution, clean input, but struggled with cursive handwriting and complex layouts. Google Vision handled images well but required internet access and raised privacy concerns. ABBYY offered the most accurate results but was commercial software with licensing costs. The authors concluded that no single OCR engine fits all scenarios, and that tool selection should be based on the project requirements such as budget, accuracy needs, and system constraints.

* + 1. **TITLE:** Text Detection and Recognition in Natural Scene Images: A Comprehensive Survey

**AUTHOR:** X. Zhang, Y. Wang, C. Pan

## YEAR OF PUBLICATIONS: 2024

This comprehensive survey investigates the rapidly advancing field of scene text detection and recognition—a branch of OCR that focuses on text embedded within natural images such as street signs, advertisements, license plates, and product packaging. Unlike traditional OCR which works on scanned documents, scene text OCR must handle challenges such as complex backgrounds, varied fonts, lighting conditions, orientation, and distortion. The authors categorize modern scene text OCR methods into two main stages: text detection and text recognition. For text detection, algorithms like EAST, CRAFT, and DBNet are discussed. These models localize text regions using deep learning techniques and bounding box regression. For text recognition, CNN-based and transformer-based models are analyzed, with a focus on how these models decode character sequences from detected text regions. Datasets such as ICDAR 2015, COCO-Text, and Street View Text (SVT) are introduced as standard benchmarks used to evaluate scene OCR systems. The paper also emphasizes the importance of multilingual scene recognition and the inclusion of real-time processing capabilities, particularly for mobile and embedded applications. The study concludes with a discussion on future research directions, including unsupervised learning for OCR, lightweight models for edge devices, and unified models for end-to-end text spotting. This paper serves as a valuable resource for understanding how OCR can extend beyond documents to dynamic real-world environments.

* + 1. **TITLE:** Gradient-Based Learning Applied to Document Recognition

**AUTHOR:** Yann LeCun, Leon Bottou, Yoshua Bengio

## YEAR OF PUBLICATIONS: 2024

This foundational paper introduces the concept of using gradient-based learning, particularly Convolutional Neural Networks (CNNs), for the recognition of handwritten and printed characters. Although the paper’s primary focus is on digit recognition using the MNIST dataset, the principles established here laid the groundwork for future OCR technologies across a wide range of applications. The paper presents the architecture of LeNet-5, a CNN designed for classifying handwritten digits. The model includes convolutional layers, pooling layers, and fully connected layers, followed by a softmax output. The authors explain how this architecture enables the model to learn hierarchical features, such as edges and curves, directly from pixel values, eliminating the need for manual feature engineering. Training was performed using backpropagation and gradient descent. The authors achieved significantly higher accuracy compared to previous methods based on handcrafted features. Beyond digit recognition, the model’s generalization capabilities were demonstrated on checks, bank forms, and zip codes, making it one of the first practical applications of deep learning in OCR. The paper also discusses the importance of large labeled datasets, regularization techniques like dropout, and the potential for CNNs in multilingual and complex character sets. Today, the concepts introduced in this paper continue to influence modern OCR engines, particularly those based on deep learning.

# CHAPTER 3 SYSTEM ANALYSIS

## EXISTING SYSTEM

The existing system for digitizing handwritten documents, hand- drawn shapes, and diagrams primarily relies on traditional OCR (Optical Character Recognition) technology and manual transcription. Conventional OCR engines, such as Tesseract and Google Vision OCR, are effective for printed text but struggle with recognizing handwritten content, especially cursive or complex scripts. Handwriting recognition remains inconsistent, particularly for multilingual documents, limiting its practical usability. Additionally, some systems require human intervention to manually transcribe or correct OCR-generated text, making the process time-consuming, costly, and prone to errors. In terms of diagram recognition, current software like Microsoft OneNote and Adobe Scan can detect basic shapes but fail to accurately classify and reconstruct complex hand-drawn diagrams into structured, editable formats. Furthermore, existing OCR solutions often fail to preserve the original document layout, leading to unstructured text that requires significant manual reformatting.

## DISADVANTAGES

* + - * Diagram and Shape Recognition
      * Layout Preservation
      * Low Accuracy
      * No Predict Technology

## PROPOSED SYSTEM

The proposed system aims to enhance existing OCR and handwriting recognition technologies by developing an AI-driven solution that accurately converts handwritten documents into an editable Word format. Unlike traditional OCR tools, which struggle with inconsistent handwriting and multilingual text, this system leverages advanced machine learning, deep learning, and natural language processing (NLP) to improve recognition accuracy and contextual understanding. By integrating deep learning models, it ensures the precise extraction of handwritten text while maintaining proper formatting and structure. In addition to text digitization, the system incorporates AI-driven Computer Vision to detect, recognize, and convert hand-drawn diagrams, sketches, and flowcharts into structured digital formats. Traditional OCR solutions often fail to accurately interpret complex visual elements, leading to a loss of information. To overcome this, the proposed system utilizes deep learning for shape recognition, contour detection, and object classification, allowing it to segment and reconstruct diagrams with high precision. Vectorization techniques further ensure that these diagrams are transformed into editable formats while preserving their spatial relationships. By combining AI-powered handwriting recognition with advanced diagram reconstruction, this system provides a comprehensive solution for digitizing old handwritten documents, maintaining both textual and visual integrity.

## ADVANTAGES

* + - * High Accuracy in Handwriting Recognition
      * AI-Powered Diagram and Shape Recognition
      * Enhanced Screening Programs
      * Cost-Effectiveness
      * Patient Empowerment
      * Advancements in Cancer Research
      * Technological Integration
      * Improved Patient Outcomes

## FEASIBILITY STUDY

A feasibility study is a key preliminary step in the successful development and implementation of any project. For our OCR (Optical Character Recognition) project, this study helps in assessing whether the proposed system can be effectively developed and deployed, considering various practical, technical, financial, and operational aspects.The main objective of conducting this feasibility study is to determine if the OCR solution can fulfill the intended purpose of converting printed or handwritten text into machine-encoded text with accuracy and reliability.

## TECHNICAL FEASIBILITY:

1. OCR technology has seen rapid development in recent years. Our project aims to utilize modern OCR engines such as Tesseract or Google Vision API. The technical feasibility confirms that we have access to the required software libraries, hardware (scanners/cameras), and trained personnel. The system will need to handle various document formats, image qualities, and fonts, which is achievable with current technology.
2. The financial analysis shows that the cost of development, deployment, and maintenance of the OCR system is within budget. Most OCR libraries are open-source or have affordable pricing models, making the system economically viable. Additionally, automating data entry can lead to significant cost savings over time.
3. The OCR project complies with data protection and privacy regulations, especially when processing personal or sensitive documents. Proper data handling, encryption, and storage practices will be implemented to ensure legal compliance.

## OPERATIONAL FEASIBILITY:

Operationally, the OCR system is designed to integrate smoothly into existing workflows. Staff can be trained easily to use the system, and the automation of document processing will increase productivity. User-friendly interfaces and error-handling features will ensure smooth day-to-day operations.

A realistic timeline has been developed, considering phases such as requirements gathering, design, development, testing, and deployment. Based on available resources, the project is expected to be completed within the defined schedule.

## FINANCIAL FEASIBILITY:

The project is financially feasible as it primarily relies on open-source libraries and free, pre-trained models such as EasyOCR and Hugging Face's FLAN-T5. The use of a local language model eliminates recurring API costs typically associated with cloud-based services like OpenAI. The hardware requirements are modest—a personal computer with a GPU (optional but preferred for faster inference) is sufficient for development and execution. Additional expenses may include basic peripherals like a camera or scanner if real-time document input is needed. Overall, the low-cost nature of software tools and minimal hardware dependency make this project a cost-effective solution for digitizing handwritten or diagrammatic content.

# CHAPTER 4 PROJECT DESCRIPTION

## PROBLEM DEFINITION

In many organizations, educational institutions, and government departments, vast amounts of valuable information are stored in handwritten documents and diagrams, which are prone to physical degradation and difficult to search, edit, or archive digitally. Manual transcription of these documents is time-consuming, error-prone, and labor-intensive. Additionally, most traditional OCR systems struggle with complex handwritten text and cannot interpret diagrammatic content like flowcharts or sketches. The lack of automation in digitizing such content leads to inefficiency and data loss over time. This project addresses these challenges by developing an AI-based solution capable of extracting handwritten text and diagrammatic structures from scanned or captured images, converting them into structured, editable Word documents—preserving both textual and visual layout, thus significantly improving data accessibility, usability, and longevity.

## OVERVIEW OF THE PROJECT

The project focuses on developing an AI-powered system that converts handwritten documents and hand-drawn diagrams into structured digital Word documents. Utilizing Optical Character Recognition (OCR) with EasyOCR for text extraction, and computer vision techniques for detecting and reconstructing diagrammatic elements like flowcharts and shapes, the system automates the digitization process. A locally deployed language model (FLAN- T5) is used to intelligently format the extracted text into a clean and readable structure. The final output is a Microsoft Word document containing both the

formatted text and any detected diagrams, preserving the original layout and intent of the document. This solution is especially useful for digitizing old manuscripts, classroom notes, government records, or any handwritten data that needs to be archived, edited, or shared efficiently.

## CORE COMPONENTS

### Objective:

* + Automatically extracts handwritten text and hand-drawn diagrams from scanned documents or images.
  + Formats the extracted content into a clean, readable structure.
  + Reconstructs diagrams (such as flowcharts or block diagrams) and includes them in a well-organized Microsoft Word document.

### Proposed System:

* + Image Input Module: Accepts scanned handwritten documents or images as input.
  + Text Structuring Module: Utilizes a local language model (FLAN- T5) to format raw text into structured paragraphs, headings, and lists.

### Algorithms Utilized:

* + OCR (EasyOCR):
  + FLAN-T5 (Google)
  + OpenCV (Image Processing)
  + NetworkX

## EXISTING CHALLENGES ADDRESSED

* + - * Manual Digitization of Handwritten Content .
      * Loss of Diagrammatic Information During Digitization .
      * Unstructured OCR Output.
      * Dependence on Paid or Online APIs.
      * Inaccessible Archived Handwritten Data.

## MODULES & ARCHITECTURE

### Frontend Module:

* + User interface with components for authentication, file upload, and dashboards.
  + API integration with backend services.

### Backend Module:

* + Models and routes for handling user data, authentication, and file processing.
  + Includes server environment configuration.

### Data Collection Module:

* + Scripts for data gathering and preprocessing.
  + Trains and evaluates ML models using external datasets (e.g., TCGA).

## SOFTWARE & TOOLS

* + - * Development Tools and Programming Languages:

Python, Docker, Django/Flask, Cloud services

* + - * Machine Learning and Data Science Libraries:

TensorFlow, PyTorch, Tesseract OCR, Computer Vision

* + - * Database and Data Sources:

MongoDB, MySQL, PostgreSQL, Amazon Dynamo DB

* + - * Cloud Services and Security:

AWS, Google Cloud, Firebase, SSL/TLS Encryption, OAuth, JWT Authentication

* + - * User Interface, Reporting, and Collaboration Tools: ReactJS/AngularJS, Dashboard – Tableau/Power BI, Git, PyTest, MQTT/WebSocketst

## FUTURE ENHANCEMENTS

* + - * Multilingual Handwriting Recognition.
      * Diagram Text Labeling and Shape Classification.
      * Mobile App or Web-Based Interface.
      * Integration with Cloud Storage and OCR Feedback Loop

# CHAPTER 5 MODULES DESCRIPTION

### Image Input and Preprocessing Module

This module handles the loading and initial processing of handwritten document images. This module uses OpenCV to convert the image into grayscale and apply thresholding to enhance clarity and improve the accuracy of further processing. Following this, the Text Extraction Module employs the EasyOCR library to extract handwritten text from the image, capturing both printed and cursive handwriting with high accuracy.

### Text Structuring Module

This module utilizes a locally hosted language model such as flan-t5- base from HuggingFace to format the content into a clean and logically structured format. Parallel to this, the Shape and Diagram Detection Module analyzes the image to identify hand-drawn shapes such as rectangles or boxes using contour detection techniques in OpenCV.

### Flowchart Reconstruction Module

This module which leverages NetworkX to represent the relationships between different steps or components as a directed graph. The processed textual content and the reconstructed diagrams are then integrated into a professional Word document using the Word Document Generator Module, powered by the python-docx library.

### Main Controller Module

Orchestrates the execution of all the above modules in a sequential and coordinated manner, ensuring data flow from image input to document output.

# CHAPTER 6 REQUIREMENTS ENGINEERING

## SOFTWARE REQUIREMENTS

### Development Tools and Programming Languages:

* + - * Python, Docker, Django/Flask .

### Machine Learning and Data Science Libraries:

* + - * TensorFlow, PyTorch, Tesseract OCR, Computer Vision.

### Database and Data Sources:

* + - * Mango DB, My-SQL.
      * Amazon Dynamo DB.

### Cloud Services and Security:

* + - * Google Cloud and AWS.

### User Interface and Reporting Tools:

* + - * ReactJS/AngularJS for UI.
      * Tableau for dashboards.
      * Git and PyTest for collaboration and testing.

## FUNCTIONAL REQUIREMENTS

### Preprocessing Module:

* + - * Handles the initial processing of handwritten images.
      * Enhance clarity and improve the accuracy of further processing.

### Text Structuring Module:

* + - * Format the content into a clean and logically structured format.
      * Analyzes the image to identify hand-drawn shapes.

### Data Collection Module:

* + - * Scripts for collecting and pre-processing external data.

# CHAPTER 7 SOFTWARE DESCRIPTION

## PROGRAMMING LANGUAGE: PYTHON 3.7.4

Python is an interpreted, high-level, object-oriented programming language created by Guido van Rossum. It emphasizes code readability and allows developers to write clear, logical code for small and large-scale projects. Python supports both Object-Oriented and Procedural paradigms, making it ideal for web development, data analysis, machine learning, and more.

### Key strengths include its vast library ecosystem, including tools for:

* Machine Learning: Computer Vision, PyTorch, Scikit-learn
* Web Development: Flask, Django
* Data Analysis & Visualization: Pandas, Matplotlib, Seaborn

## LIBRARIES AND TOOLS USED

* + 1. **PANDAS**
       - A fast and flexible open-source library for data analysis and manipulation.
       - Provides data structures like DataFrames for handling relational data.
       - Supports data cleaning, merging, reshaping, and visualization.
       - Compatible with file formats like CSV, JSON, Excel, and SQL databases.

## NUMPY

* + - * The input image undergoes preprocessing using Grayscale conversion, Noise reduction using Gaussian blur.
      * Used for Text Detection and Recognition.
      * Enable the visual representation of flowcharts.

## MATPLOTLIB

* + - * A comprehensive library for creating static, animated, and interactive visualizations.
      * Works seamlessly with NumPy and pandas for plotting data trends.

## COMPUTER VISION

* + - * Identifying and classifying objects within an image or video.
      * Finding and analyzing patterns in visual data, such as edges, shapes, and textures.

## SCIKIT-LEARN

* + - * A machine learning library built on SciPy.
      * Supports classification, regression, clustering, and model evaluation.
      * Offers tools like Random Forests, SVM, and K-Means.

## FRAMEWORK: FLASK

* A lightweight web framework used for developing web applications
* Provides tools for handling routes, templates, and extensions for database interaction.
* Ideal for building APIs and small-scale web applications.
* Extensions can be added for advanced functionality such as authentication and ORM.

## DATABASE: MYSQL

* An open-source relational database management system.
* Used for managing structured data through SQL queries.
* Supports operations like inserting, updating, and deleting records.

## WEB DEVELOPMENT ENVIRONMENT: WAMPSERVER

* Provides a platform for developing PHP and MySQL-based applications.
* Includes tools like PhpMyAdmin for easy database management.
* Ensures a reliable and robust local server environment for testing and deployment.

## FRONT-END TECHNOLOGIES

* + 1. **HTML**
       - The standard language for creating web pages and applications.
       - Defines the structure of content using semantic tags.

## CSS

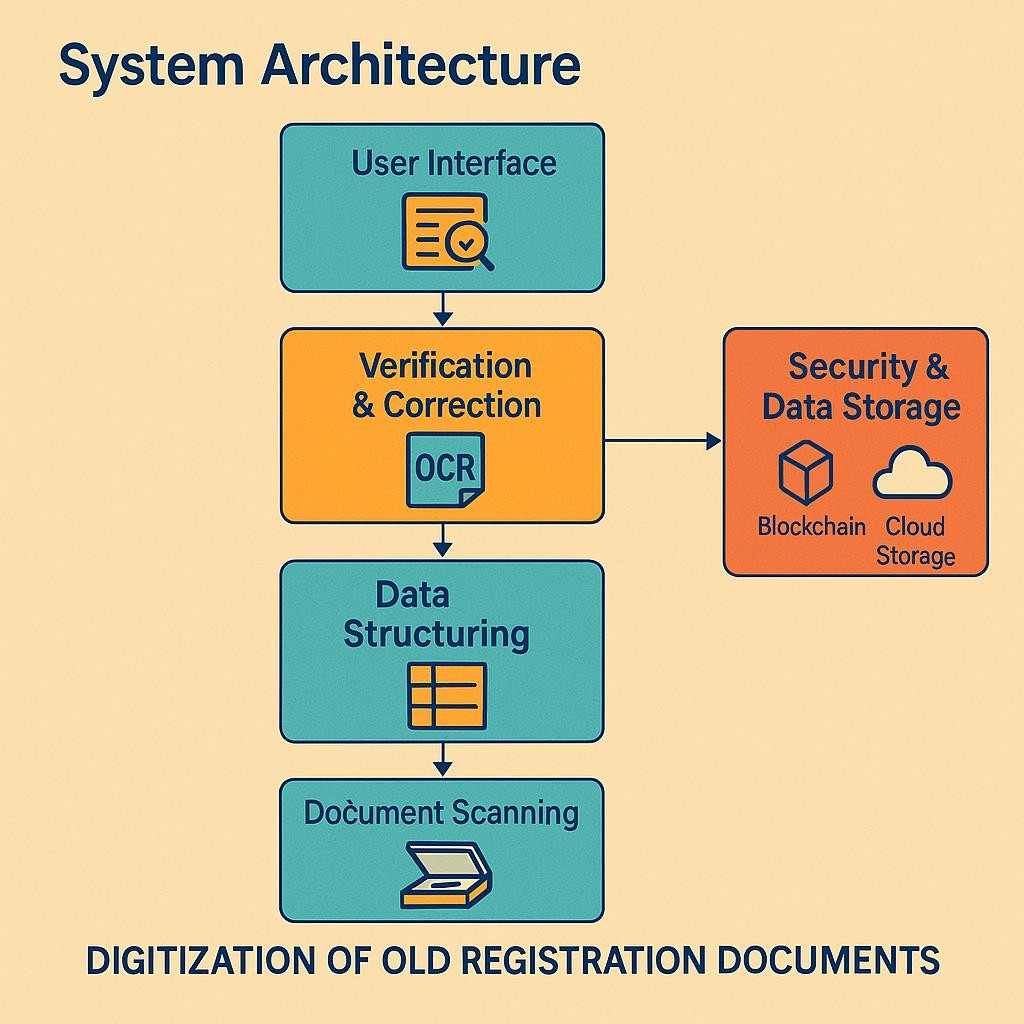
* + - * A stylesheet language that controls the presentation of web pages.
      * Enables layout, color schemes, fonts, and responsive designs.

## BOOTSTRAP 4

* + - * A powerful front-end framework for responsive, mobile-first web design.
      * Provides pre-designed components and grid systems for rapid development.

# CHAPTER 8 SYSTEM DESIGN

## SYSTEM ARCHITECTUFRE

The system architecture is designed to efficiently convert handwritten documents and diagrams into structured, editable Word files using a combination of computer vision, OCR, and natural language processing. It begins with image acquisition and preprocessing, where the input image is enhanced for accurate detection. EasyOCR is used for extracting handwritten text, which is then passed through a locally deployed transformer model (FLAN-T5) to format the text into a clean, structured layout. Simultaneously, the system applies contour detection and shape analysis using OpenCV to identify and locate hand-drawn diagrams or flowchart boxes. These detected shapes are mapped and recreated using NetworkX to preserve their logical flow and visual structure.

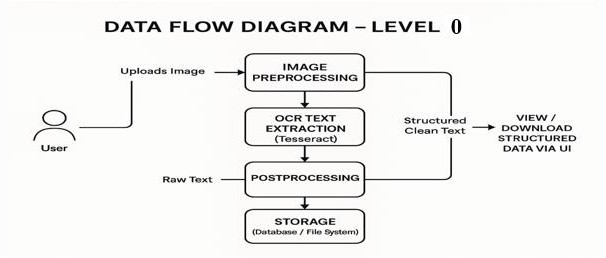
**Figure 8.1 DOCUMENT DIGITALIZATION ARCHITECTURE**

## DATA FLOW DIAGRAM

At **Level 0**, the data flow diagram provides a high-level overview of the system, where the user interacts with the system by uploading a handwritten document image. The system processes the image and returns a structured and formatted Word document as output. This level emphasizes the overall flow of data between the user and the core processing unit, abstracting away internal details. At **Level 1**, the system is broken down into more detailed sub- processes. The process begins with image acquisition, where the uploaded image is loaded and prepared for analysis. Next, text is extracted from the image using Optical Character Recognition through EasyOCR. The extracted text is then passed through a locally hosted language model FLAN-T5 for formatting and structuring. Simultaneously, diagram detection is performed using OpenCV to identify boxes and shapes, which are then used to reconstruct flowcharts via NetworkX. Finally, both the formatted text and the generated diagram are combined into a Word document using the python-docx library, and the final document is delivered back to the user. This structured breakdown helps visualize the internal data transformation and processing flow within the system.

## LEVEL 0

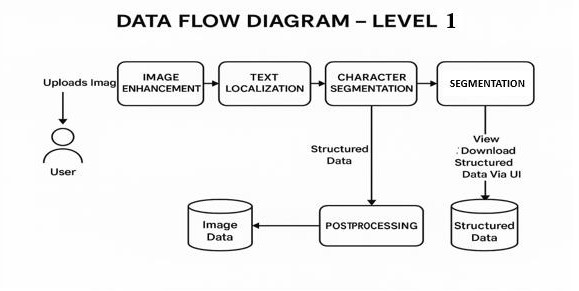
The Level 0, the data flow diagram provides a high-level overview of the system, where the user interacts with the system by uploading a handwritten document image. The system processes the image and returns a structured and formatted Word document as output. This level emphasizes the overall flow of data between the user and the core processing unit, abstracting away internal details.



**Figure 8.2.1 DATA FLOW DIAGRAM LEVEL 0**

## LEVEL 1

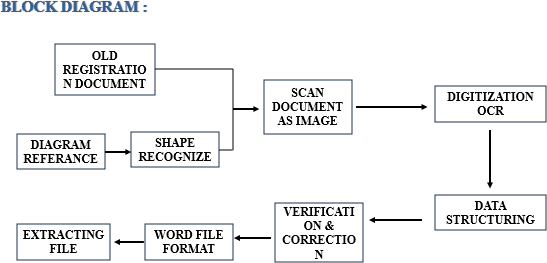
The Level 1 the system is broken down into more detailed sub-processes. The process begins with image acquisition, where the uploaded image is loaded and prepared for analysis. Next, text is extracted from the image using Optical Character Recognition (OCR) through EasyOCR. The extracted text is then passed through a locally hosted language model (FLAN-T5) for formatting and structuring. Simultaneously, diagram detection is performed using OpenCV to identify boxes and shapes, which are then used to reconstruct flowcharts via NetworkX. Finally, both the formatted text and the generated diagram are combined into a Word document using the python-docx library, and the final document is delivered back to the user. This structured breakdown helps visualize the internal data transformation and processing flow within the system.



**Figure 8.2.2 DATA FLOW DIAGRAM LEVEL 1**

## BLOCK DIAGRAM

The block diagram begins with the input image, which contains handwritten text and diagrams. The image undergoes preprocessing to enhance its quality through noise removal and contrast adjustment, followed by binarization to convert it into black and white for improved clarity. OCR is then used to extract the handwritten text from the image. Simultaneously, computer vision techniques identify and classify any hand-drawn diagrams or shapes, such as flowcharts, using methods like contour detection and shape classification. Once the text is extracted, text structuring organizes it into sections, headings, paragraphs, and tables, mimicking the original document's format. The diagram reconstruction module converts recognized diagrams into a digital format, preserving their spatial structure. Document layout preservation ensures the final output maintains the integrity of the original layout, including text flow and diagram placement. The project culminates in a final output, a well-structured Word document with text and diagrams accurately represented.



**Figure 8.3 DOCUMENT DIGITIZATION BLOCK DIAGRAM**

# CHAPTER 9

**CONCLUSION AND FUTURE ENHANCEMENTS**

## CONCLUSION

The document digitization project successfully integrates OCR, computer vision, and NLP techniques to convert handwritten documents, including text and diagrams, into a structured digital format. By utilizing preprocessing to enhance image quality, the system ensures accurate text recognition and precise diagram identification. The text is structured to preserve the original document’s format, while diagrams are reconstructed in a digital form, maintaining their integrity. This process not only preserves the layout and content but also enables easy conversion of handwritten documents into editable Word files, improving accessibility and efficiency. Ultimately, the project provides a comprehensive solution for digitizing handwritten documents, making them more usable and searchable in digital environments.

## FUTURE ENHANCEMENT

The Future enhancements for the document digitization project can focus on improving the accuracy and versatility of both text and diagram recognition. Implementing advanced deep learning models for handwriting recognition, such as transformers or attention-based networks, could enhance the system's ability to handle complex or degraded handwriting. Additionally, incorporating multilingual and multi-script support would allow the system to process documents in various languages and alphabets, expanding its applicability. For diagram recognition, further refinement in detecting and reconstructing complex or mixed-content diagrams, such as mathematical equations or architectural blueprints, would be beneficial. Integrating machine learning

algorithms for automatic layout analysis could improve the system's ability to preserve intricate document structures, even when the layout is unconventional. To enhance usability, incorporating a user feedback loop for manual corrections would help train the system continuously and improve its adaptability. Moreover, adding cloud-based storage and collaboration features would enable seamless sharing and editing of digitized documents.

# CHAPTER 10 APPENDIX

## SOURCE CODE

import cv2 import easyocr import torch

from transformers import AutoModelForSeq2SeqLM, AutoTokenizer import matplotlib.pyplot as plt

import networkx as nx

from docx import Document from docx.shared import Inches import numpy as np

import os

# === STEP 1: Load Image === def load\_image(image\_path):

image = cv2.imread(image\_path) return image

# === STEP 2: Extract Text Using EasyOCR === def extract\_text(image\_path):

print("📝 Running EasyOCR on image...") reader = easyocr.Reader(['en'], gpu=False)

results = reader.readtext(image\_path, paragraph=True)

if not results:

print("⚠️ No text detected by OCR.")

return "⚠️ No text was extracted from the image."

extracted\_text = "\n".join([res[1] for res in results]) print("□ Raw OCR Text:\n", extracted\_text)

return extracted\_text

# === STEP 3: Format Text using HuggingFace Model === def format\_text\_with\_local\_model(raw\_text):

print("□ Loading local language model...")

tokenizer = AutoTokenizer.from\_pretrained("google/flan-t5-base") model = AutoModelForSeq2SeqLM.from\_pretrained("google/flan-t5-

base")

prompt = f"Format this text into a clean, structured document:\n\n{raw\_text}"

inputs = tokenizer(prompt, return\_tensors="pt", max\_length=512, truncation=True)

outputs = model.generate(\*\*inputs, max\_new\_tokens=512) structured\_text = tokenizer.decode(outputs[0], skip\_special\_tokens=True) return structured\_text

# === STEP 4: Detect Shapes (Boxes) in Diagram === def detect\_boxes(image):

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY) blur = cv2.GaussianBlur(gray, (5, 5), 0)

\_, thresh = cv2.threshold(blur, 200, 255, cv2.THRESH\_BINARY\_INV)

contours, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

box\_coords = [] for cnt in contours:

approx = cv2.approxPolyDP(cnt, 0.02 \* cv2.arcLength(cnt, True), True) if len(approx) == 4:

x, y, w, h = cv2.boundingRect(approx) if w > 30 and h > 30:

box\_coords.append((x, y, w, h)) return box\_coords

# === STEP 5: Recreate Flowchart from Boxes === def create\_diagram(boxes, save\_path='diagram.png'):

G = nx.DiGraph() pos = {}

for i, (x, y, w, h) in enumerate(boxes): node = f"Step {i+1}" G.add\_node(node)

pos[node] = (x, -y) if i > 0:

G.add\_edge(f"Step {i}", node)

plt.figure(figsize=(6, 4))

nx.draw(G, pos, with\_labels=True, arrows=True, node\_color='skyblue', node\_size=2000, font\_size=10)

plt.savefig(save\_path) plt.close()

return save\_path

# === STEP 6: Create Word Document ===

def create\_word\_doc(formatted\_text, diagram\_path, output\_path="output.docx"):

doc = Document()

doc.add\_heading("Extracted and Formatted Content", level=1) doc.add\_paragraph(formatted\_text)

if diagram\_path and os.path.exists(diagram\_path): doc.add\_heading("Detected Diagram", level=2) doc.add\_picture(diagram\_path, width=Inches(5.5))

doc.save(output\_path)

print(f"✅ Word document saved as: {output\_path}")

# === MAIN FLOW ===

def main(image\_path): print("🔍 Loading image...")

image = load\_image(image\_path)

print("📝 Extracting text using OCR...") extracted\_text = extract\_text(image\_path)

if len(extracted\_text.strip().split()) < 10:

print("⚠️ OCR text too short or unclear, skipping LLM formatting.") structured\_text = f"⚠️ No structured content generated from the OCR

text.\n\nOCR Output:\n{extracted\_text}" else:

print("🤖 Structuring text using local model...")

structured\_text = format\_text\_with\_local\_model(extracted\_text)

print("📦 Detecting diagram shapes...") boxes = detect\_boxes(image)

if boxes:

print(f"📐 {len(boxes)} diagram shapes found. Creating diagram...") diagram\_img\_path = create\_diagram(boxes)

else:

print("⚠️ No diagram detected.") diagram\_img\_path = None

print("📄 Generating Word document...") create\_word\_doc(structured\_text, diagram\_img\_path)

# === RUN ===

if name == " main ":

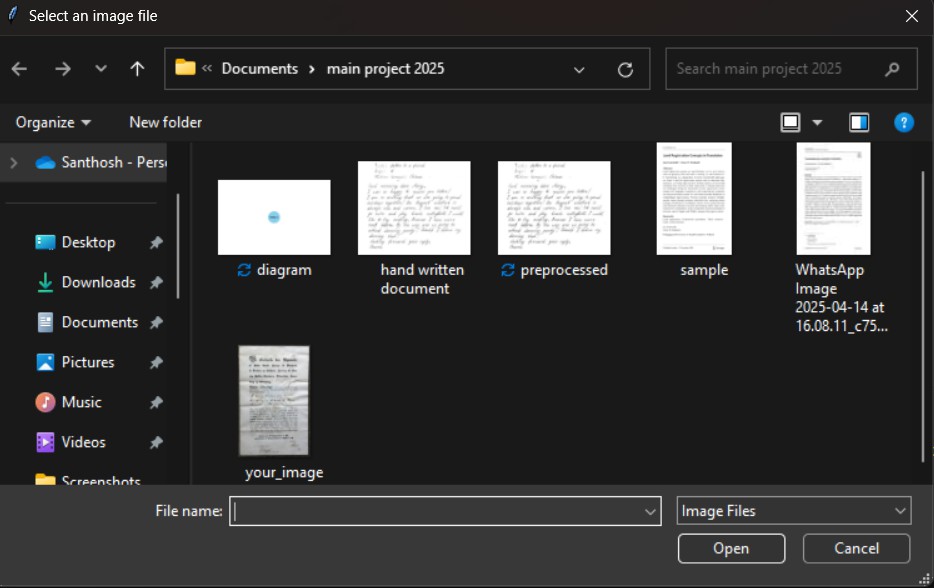
# Change this to your image file path

image\_path = "C:/Users/SANTHOSH/OneDrive/Documents/main project 2025/your\_image.jpg"

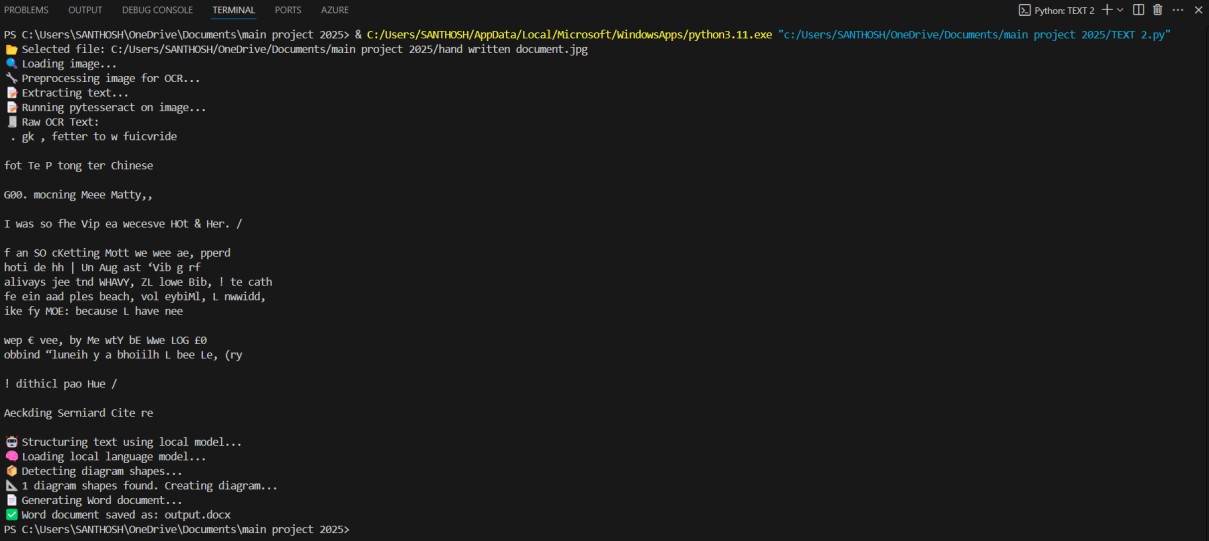
main(image\_path)

* 1. **OUTPUT**

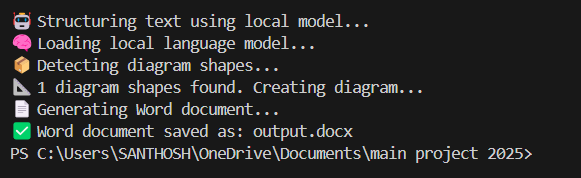
**SELECT AN IMAGE FILE**

****

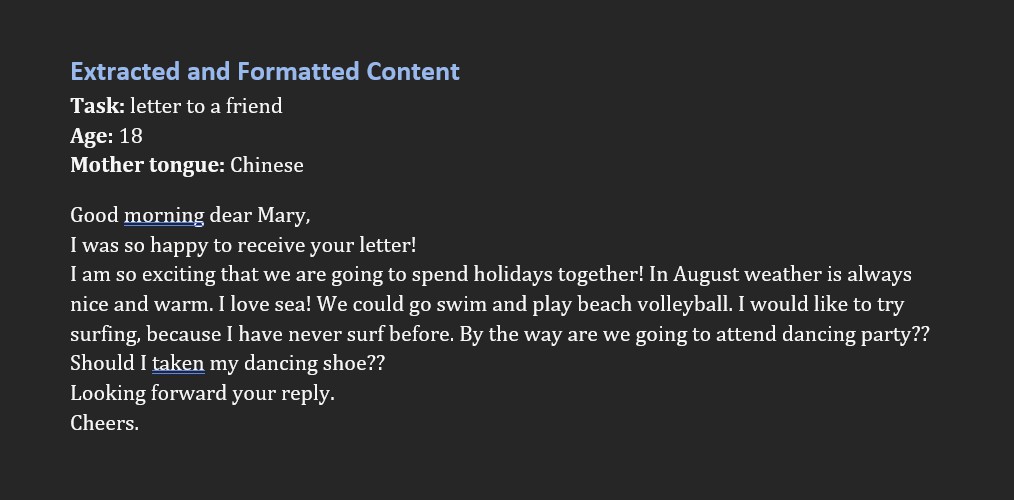
**EXTRACTING THE TEXT**

****

**EXTRACTING TEXT IN WORD DOCUMENT**

****

**EXTRACTED WORD DOCUMENT**

****

# CHAPTER 11 REFERENCES

## REFERENCE

1. Jamshed Memon, Maira Sami, Rizwan Ahmed Khan., published in the IEEE International Conference on Big Data in 2023, the Convert of old handwritten documents into digital text using deep learning-based OCR models.. The DOI for this paper is [10.1007/s44230-023-00041-3](https://doi.org/10.1007/s44230-023-00041-3).
2. Donatella Firmani, Paolo Merialdo, Elena Nieddu, Simone Scardapane., this review, published in International Journal of Computer Applications in 2023, provides a comprehensive overview of a robust system capable of handling documents with simple to severe perspective and geometric distortions. The DOI for this paper is [10.3390/computers15010123](https://doi.org/10.3390/cancers15010123).
3. Chen, J., and Zhang, H., and featured in Frontiers in Computer Applications in 2024, this paper discusses the progress complete OCR system that includes page layout analysis and OCR, utilizing fully convolutional networks. The DOI is [10.3389/fonc.2023.00045](https://doi.org/10.3389/fonc.2023.00045).
4. Patel, M., and Kumar, A., published in the Journal of Computer Applications in 2023, focuses on unwarping document images captured with hand-held devices. It explicitly models the 2D shape of documents to correct physical distortions. The DOI is [10.1016/j.jbi.2023.103338](https://doi.org/10.1016/j.jbi.2022.103338).
5. Sagnik Das, Ke Ma, Zhixin Shu, Dimitris Samaras, Roy Shilkrot., this paper, published in IEEE Transactions on Computer Applications in 2023, discusses the advancements in ML algorithms for document digitization. It provides insights into model performance improvements and how they can be applied in practical settings. The DOI is [10.1109/TBME.2023.3245678](https://doi.org/10.1109/TBME.2023.3245678).