**OPTICAL CHARACTER RECOGNITION**

**PROJECT STAGE 1 REPORT**

**(20CA3503)**

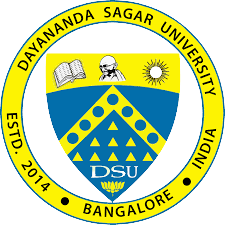
**2022-2023**

**SUBMITTED BY**

**NANDINI GARG (ENG20CA0023)**

**SHREYA JAISWAL (ENG20CA0042)**

**BACHELOR OF COMPUTER APPLICATION**



**DEPARTMENT OF COMPUTER APPLICATIONS**

**SCHOOL OF ENGINEERING**

**DAYANANDA SAGAR UNIVERSITY**

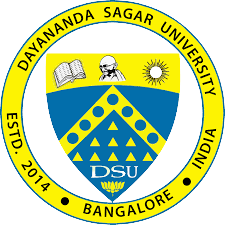
**KUDLU GATE, HOSUR ROAD, BANGALORE-560068**

**DECEMBER 2022**

**DAYANANDA SAGAR UNIVERSITY**

**KUDLU GATE, HOSUR ROAD, BANGALORE – 560068**

**DEPARTMENT OF COMPUTER APPLICATION**



**BONAFIDE CERTIFICATE**

This is to certify that the project work entitled **“OPTICAL CHARACTER RECOGNITION”** is a bonafide record of the workcarried out by **Nandini Garg (ENG20CA0023), Shreya Jaiswal (ENG20CA0042)** at **Dayananda Sagar University,** during the year **2022-2023.**

The project report has been approved as it satisfies the academic requirements in respect

of project work prescribed for the said degree.

**GUIDE CHAIRPERSON**

**Dr. Vasanthi Kumari P Dr. Vasanthi Kumari P**

Chairperson, Chairperson,

Department of Computer Applications, Department of Computer Applications,

School of Engineering, School of Engineering,

Dayananda Sagar University. Dayananda Sagar University.

**Project viva voice held on - \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Signature of the external examiner**



**ACKNOWLEDGEMENT**

We take this opportunity to express our sincere gratitude and respect to **Dayananda Sagar University,** Bangalore for providing us a platform to pursue our studies and carry out our second-year project. We have an immense pleasure in expressing our deep sense of gratitude to **Dr. Vasanthi Kumari P, Chairperson,** Dayananda Sagar University, Bangalore, for her exemplary guidance, valuable suggestions, expert advice, and encouragement to pursue this project work.

This project helped us in understanding the various parameters which are involved in the development of an OCR system.

We would like to thank **Dr. Udaya Kumar Reddy K R, Dean,** Dayananda Sagar University, Bangalore, who has been a constant support and encouragement throughout the course of this project.

We also extend our thanks to all the faculty of Computer Applications who directly or indirectly encouraged us. Finally, we would like to thank our parents and friends for all their moral support they have given us during the completion of this work.

Lastly, to the almighty, for showering their blessings and to many more, whom we didn’t mention here.

**Nandini Garg (ENG20CA0023)**

**------------------------------------------**

**Shreya Jaiswal (ENG20CA0042)**

**------------------------------------------**



**ABSTRACT**

In many different fields, there is a high demand for storing information to a computer storage disk from the data available in printed or handwritten documents or images to later re-utilize this information by means of computers. One simple way to store information to a computer system from these printed documents could be first to scan the documents and then store them as image files. But to re-utilize this information, it would be very difficult to read or query text or other information from these image files. Therefore, a technique to automatically retrieve and store information, in particular text, from image files is needed. Optical character recognition is an active research area that attempts to develop a computer system with the ability to extract and process text from images automatically. The objective of OCR is to achieve modification or conversion of any form of text or text-containing documents such as handwritten text, printed or scanned text images, into an editable digital format for deeper and further processing. Therefore, OCR enables a machine to automatically recognize text in such documents. Some major challenges need to be recognized and handled in order to achieve a successful automation. The font characteristics of the characters in paper documents and quality of images are only some of the recent challenges. Due to these challenges, characters sometimes may not be recognized correctly by computer system.  In this report, we discussed the various stages in text recognition, handwritten OCR systems classification according to the text type as well as application oriented recent research in OCR. Therefore, this discussion provides a very comprehensive review of the state-of-the-art of the field.



**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S.N.** | **TOPICS** | **PAGE NO.** |
|  | ACKNOWLEDGEMENT | i |
|  | ABSTRACT | ii |
|  | TABLE OF CONTENT | iii |
|  | LIST OF FIGURES | iv |
|  | LIST OF TABLES | iiv |
|  |  |  |
| **1** | **CHAPTER:1 – INTRODUCTION** | **9** |
|  | 1.1 General | 10 |
|  | 1.2 Objective | 10 |
|  | 1.3 Existing system | 10 |
|  | 1.4 Drawback of existing system | 10 |
|  | 1.5 Problem statement | 11 |
|  | 1.6 Proposed system | 11 |
|  | 1.7 Need for the project | 11 |
|  | 1.8 Scope | 11 |
| **2** | **CHAPTER:2 – LITERATURE SURVEY** | **12** |
|  | 2.1 Optical Character Recognition using Tesseract and Classification | 13 |
|  | 2.2 A Detailed Analysis of Optical Character Recognition Technology | 13 |
|  | 2.3 Computer Vision and Deep Learning Resource Guide | 13 |
|  | 2.4 Multi-Lingual Optical Character Recognition System Using the Reinforcement Learning of Character Segmenter | 13 |
|  | 2.5 Multi-Lingual Optical Character Recognition System Using the Reinforcement Learning of Character Segmenter | 13 |
|  | 2.6 An overview of character recognition focused on off-line handwriting | 14 |
|  | 2.7 Optical Character Recognition by Open-source OCR Tool Tesseract: A Case Study | 14 |
|  | 2.8 A Survey on various Optical Character Recognition Techniques | 14 |
|  | 2.9 Summary of the literature survey | 15 |
| **3** | **CHAPTER:3 – SYSTEM DESIGN** | **16** |
|  | 3.1 System model | 17 |
|  | 3.1.1 Applications of OCR | 17 |
|  | 3.2 Functional requirement | 18 |
|  | 3.2.1 Modules and their functionalities | 19 |
|  | 3.3 System Architecture | 20 |
|  | 3.4 UML Diagrams | 20 |
| 4 | **CHAPTER: 4 – IMPLEMENTATION** | 28 |
|  | 4.1 Tools and environment | 29 |
|  | 4.2 Domain and Languages | 29 |
|  | 4.3 IDE and server | 30 |
|  | 4.4 Software requirement | 31 |
|  | 4.5 Hardware requirement | 31 |
| 5 | **CHAPTER: 5 - TESTING** | 32 |
|  | 5.1 Types of testing | 33 |
|  | 5.2 Unit testing | 34 |
|  | 5.3 Integration testing | 34 |
|  | 5.4 Acceptance testing | 34 |
| 6 | **CHAPTER: 6- RESULTS** | 35 |
|  | 6.1 importing module | 36 |
|  | 6.2 importing all images from path | 36 |
|  | 6.3 importing the downloaded languages | 37 |
|  | 6.4 extracting text from an image: simple | 37 |
|  | 6.5 extract text from image: specified language | 38 |
|  | 6.6 extract text from image: timeout extraction | 38 |
|  | 6.7 Get bounding box estimates | 39 |
|  | 6.8 Verbose data | 40 |
|  | 6.9 Information about orientation and script detection | 41 |
|  | 6.10 Converting to different file formats | 41 |
|  | 6.11 Extracting text from PDF | 42 |
|  | 6.12 Extract text from multiple images | 43 |
|  | 6.13 converting image text to audio | 44 |
| 7 | **CHAPTER: 7- CONCLUSION AND FUTURE ENHANCEMENTS** | 45 |
|  | 7.1 Conclusion | 46 |
|  | 7.2 Future enhancement | 46 |
| 8 | **CHAPTER: 8 – REFERENCES** | 47 |

**LIST OF FIGURES**



|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **NAME** | **PAGE NO.** |
| 3.1.1 | CLASSIC OCR PROCEDURE MODEL | 17 |
| 3.3.1 | ARCHITECTURE OF OCR SYSTEM | 20 |
| 3.4.1 | DATA FLOW DIAGRAM | 21 |
| 3.4.2 | USE CASE DIAGRAM | 21 |
| 3.4.3 | CLASS DIAGRAM | 22 |
| 3.4.4 | SEQUENCE DIAGRAM | 23 |
| 3.4.5 | ACTIVITY DIAGRAM | 24 |
| 3.4.6 | COMPONENT DIAGRAM | 25 |
| 3.4.7 | DEPLOYMENT DIAGRAM | 26 |
| 6.2.1 | IMPORTING IMAGES FROM PATH | 35 |
| 6.2.2 | IMAGES IN PATH | 35 |
| 6.3.1 | IMPORTING LANGUAGES FROM FILE | 36 |
| 6.4.1 | SIMPLE IMAGE EXTRACTION | 36 |
| 6.5.1 | TEXT EXTRACTION FROM SPECIFIC LANGUAGE | 37 |
| 6.6.1 | TIMEOUT TEXT EXTRACTION | 38 |
| 6.7.1 | BOUNDING BOX ESTIMATES | 38 |
| 6.7.2 | BOUNDING BOX OUTPUT | 39 |
| 6.8.1 | VERBOSE DATA EXTRACTION | 39 |
| 6.9.1 | ORIENTAION AND SCRIPT DETECTION | 40 |
| 6.10.1 | DIFFERENT FILE FORMATS | 41 |
| 6.11.1 | TEXT FROM PDF EXTRACTION | 41 |
| 6.12.1 | TEXT FROM MULTIPLE IMAGE EXTRACTION | 42 |
| 6.12.2 | MULTIPLE IMAGES IN FILE | 42 |
| 6.13.1 | IMAGE TEXT TO AUDIO | 43 |

**LIST OF TABLES**



|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **NAME** | **PAGE NO.** |
| 2.9.1 | Literature Survey | 15 |

**CHAPTER 1:**

**INTRODUCTION**

* 1. **GENERAL:**

Optical Character Recognition is the technology used for converting the transcribed, handwritten or any printed text documents such as scanned pages, images taken by phone or documents into the text data that can be edited and reused. In other words, OCR takes a look on the photo of the text document (therefore it is called as "optical" process) and then recognizes the different alphabets, numbers or any other characters. This sub process is called as character recognition, which is used to fetch the characters from the image, and then these characters will be converted to text sentences for further use. This mainly aims to reduce the human workload, and it achieves the same as it is handy and it also saves the time as it provides all the text that the user was supposed to be retyping. Our OCR is capable of giving out the output text quickly, but the handwritten text recognition takes little longer. Generally, the process of OCR has three stages, that are: Process (Scan) the image document, Recognize the text data and then save it into any convenient format or display it directly to the user for further use.

**1.2 OBJECTIVE:**

**Our project was made-up having the following key points in mind. It mainly aims to:**

1.To allow extraction of the information that a user wants from the paper document and using it wherever it is needed. This leads to reduction or sometimes eliminating the work of costly data entry.

2. To enable a way in which processing of the documents will lead to eliminate the human touches and therefore dramatically reducing the process time and the cost.

3. To take an image as input and give the editable text to the user which is recognized from the image document.

**1.3 EXISTING SYSTEM:**

In the running world there is a growing demand for the users to convert the printed documents in to electronic documents for maintaining the security of their data. Hence the basic OCR system was invented to convert the data available on papers in to computer processable documents, so that the documents can be editable and reusable.

The existing system/the previous system of OCR on a grid infrastructure is just OCR without grid functionality. That is the existing system deals with the homogeneous character recognition or character recognition of single languages.

**1.4 DRAWBACK OF EXISTING SYSTEM**

The drawback in the early OCR systems is that they only have the capability to convert and recognize only the documents of english or a specific language only. That is, the older OCR system is uni-lingual.

**1.5 PROBLEM STATEMENT:**

The problem here is for the software systems to recognize characters in computer system when information is scanned through paper documents as we know that we have number of newspapers and books which are in printed format related to different subjects. Whenever we scan the documents through the scanner, the documents are stored as images such as jpeg, gif etc, in the computer system. These images cannot be read or edited by the user. But to reuse this information it is very difficult to read the individual contents and searching the contents form these documents line-by-line and word-by-word. These days there is a huge demand in “storing the information available in these paper documents in to a computer storage disk and then later editing or reusing this information by searching process”.

**1.6 PROPOSED SYSTEM:**

Our proposed system is OCR which can extract text from different types of image formats like jpg, png and pdf too and can recognize mostly all languages. This proposed system also includes modules like image to audio conversion, giving the dimensions of images, it does alphabet screening, giving coordinates for images, and providing bounded rectangles for specific identification. Also, our project includes the conversion of images into PDF, HOCR and XML formats. This project can extract text from pdf also and display the pages in pdf. It also has the ability to extract the whole folder of images

**1.7 NEED FOR THE PROJECT:**

The need for this project is that it overcomes the drawback of the existing system, it supports multiple functionalities such as editing and searching. It also adds benefits by providing heterogeneous character recognition.

**1.8 SCOPE & STRENGTH OF PROJECT:**

The scope of our product Optical Character Recognition is to provide an efficient and enhanced tool for the users to perform Document Image Analysis, document processing by reading and recognizing the characters in research, academic, governmental and business organizations that are having large pool of documented, scanned images. Irrespective of the size of documents and the type of characters in documents, the product is recognizing them, searching them and processing them faster according to the needs of the environment

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 Optical Character Recognition using Tesseract and Classification**

*(2021 International Conference on Emerging Smart Computing and Informatics (ESCI) AISSMS Institute of Information Technology, Pune, India. Mar 5-7, 2021)*

**Authors: Saurabh Dome, Asha P Sathe, Department of Computer Engineering Army Institution of Technology**

In this paper, OCR WebApp has been experimentally proven to work satisfactorily by consistently providing higher accuracy. This the paper presents the design and procedure of the OCR WebApp, which consists of three sections that are: Image-to-Text, Real-time OCR (using webcam), and Handwritten Text Recognition. In this project, OCR uses Tesseract as an engine to display the text to the user and HTR uses a Deep learning model to classify the letters and display them to the user.

**2.2 A Detailed Analysis of Optical Character Recognition Technology**

*(International Journal of Applied Mathematics, Electronics and Computers)*

**Authors: Karez Abdulwahhab Hamad \*1, Mehmet Kaya**

In this paper, Numerous algorithms, methods and techniques have been proposed to optical character recognition in scene imagery. They highlighted that for designing any application related to the OCR, one must pay great attention to each phase to obtain high accurate character recognition rate, but still, we cannot propose comprehensive algorithms for each phase because it depends upon datasets, application specifics, and parameter specifics.

**2.3 Multi-Lingual Optical Character Recognition System Using the Reinforcement Learning of Character Segmenter**

***(****From IEEE)*

**Authors:** [**Jaewoo Park**](https://ieeexplore.ieee.org/author/37086516863)**;** [**Eunji Lee**](https://ieeexplore.ieee.org/author/37673934600)**;** [**Yoonsik Kim**](https://ieeexplore.ieee.org/author/37086110381)**;** [**Isaac Kang**](https://ieeexplore.ieee.org/author/37088447122)

In this document, they presented a new multi-lingual Optical Character Recognition (OCR) system for scanned documents. In the case of Latin characters, current open-source systems such as Tesseract provide very high accuracy. However, the accuracy of the multi-lingual documents, including Asian characters, is usually lower than that for Latin-only documents. They adopted the REINFORCE algorithm and train the segment.

**2.4 Computer Vision and Deep Learning Resource Guide**

*(From pyimagesearch)*

**Authors: Dr. Adrian Rosebrock**

In this guide, we learned about the python libraries and packages with modules. A blog for OCR is given by the author.

**2.5 Handwritten Character Recognition to Obtain Editable Text**

*(From IEEE)*

**Authors:** [**Vaibhav. V. Mainkar**](https://ieeexplore.ieee.org/author/37088384031)**;** [**Jyoti A. Katkar**](https://ieeexplore.ieee.org/author/37088457941)**;** [**Ajinkya B. Upade**](https://ieeexplore.ieee.org/author/37088459151)**;** [**Poonam**](https://ieeexplore.ieee.org/author/37088458189)

In this document, they developed an android application for character recognition to read the text from an image is a big area of research. This system uses the android phone to capture the image of the document and further steps are done by OCR. The main challenge is to recognize the characters from different styles of handwriting. This system offers 90% accuracy for handwritten documents and gives the easiest way to edit or share the recognized data.

**2.6 An overview of character recognition focused on off-line handwriting**

*(June 2001,* [*IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)*](https://www.researchgate.net/journal/IEEE-Transactions-on-Systems-Man-and-Cybernetics-Part-C-Applications-and-Reviews-1094-6977) *31(2):216 - 233 )*

**Authors: Nafiz Arica & Fatos Tunay Yarman Vural**

This paper serves as a guide and update for readers working in the CR area. First, the historical evolution of CR systems is presented. Then, the available CR techniques, with their superiorities and weaknesses, are reviewed. Finally, the current status of CR is discussed and directions for future research are suggested. Special attention is given to off-line handwriting recognition, since this area requires more research in order to reach the ultimate goal of machine simulation of human reading.

**2.7 Optical Character Recognition by Open-source OCR Tool Tesseract: A Case Study**

*(International Journal Of Computer Applications- December 2022 Edition)*

**Authors: Chirag Patel, Atul Patel Dharmendra Patel**

Optical character recognition (OCR) method has been used in converting printed text into editable text. OCR is very useful and popular method in various applications. Accuracy of OCR can be dependent on text pre-processing and segmentation algorithms. Sometimes it is difficult to retrieve text from the image because of different size, style, orientation, complex background of image etc. We begin this paper with an introduction of Optical Character Recognition (OCR) method, History of Open-Source OCR tool Tesseract, architecture of it and experiment result of OCR performed by Tesseract on different kinds images are discussed. We conclude this paper by comparative study of this tool with other commercial OCR tool Transym OCR by considering vehicle number plate as input.

**2.8 A Survey on various Optical Character Recognition Techniques**

*(From IEEE -* [*2018 Conference on Emerging Devices and Smart Systems (ICEDSS)*](https://ieeexplore.ieee.org/xpl/conhome/8520539/proceeding)*)*

**Authors:** [**Abin M Sabu**](https://ieeexplore.ieee.org/author/37086525166)**;** [**Anto Sahaya Das**](https://ieeexplore.ieee.org/author/37086526793)

This paper gives various optical character recognition techniques that is used for various character recognition. Optical character recognition is a technique in which a scanned images or handwritten notes are converted into digital format. Optical Character Recognition consists of various stages includes pre-processing, Classification, Post Acquisition, Pre-Level processing, Segmented Processing, Post-Level processing, Feature Extraction.

**2.9 Summary of Literature Survey**

|  |  |  |
| --- | --- | --- |
| **S. NO.** | **TITLE** | **KEY FEATURES** |
| **1.** | **Optical Character Recognition using Tesseract and Classification**  *(2021 International Conference on Emerging Smart Computing and Informatics (ESCI) AISSMS Institute of Information Technology, Pune, India. Mar 5-7, 2021)* | This paper presents the design and procedure of the OCR WebApp, which consists of three sections that are: Image-to-Text, Real-time OCR, and Handwritten Text Recognition. |
| **2.** | **A Detailed Analysis of Optical Character Recognition Technology**  *(International Journal of Applied Mathematics, Electronics and Computers)* | In this paper, Numerous algorithms, methods and techniques have been proposed to optical character recognition in scene imagery. |
| **3.** | **Computer Vision and Deep Learning Resource Guide**  *(From pyimagesearch)* | Learned about the python libraries and packages with modules. |
| **4.** | **Multi-Lingual Optical Character Recognition System Using the Reinforcement Learning of Character Segmenter**  ***(****From IEEE)* | In this document, they presented a new multi-lingual Optical Character Recognition (OCR) system for scanned documents. |
| **5.** | **Handwritten Character Recognition to Obtain Editable Text**  *(From IEEE)* | In this document, they developed an android application for character recognition to read the text from an image is a big area of research. |
| **6.** | **An overview of character recognition focused on off-line handwriting**  *(June 2001,* [*IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)*](https://www.researchgate.net/journal/IEEE-Transactions-on-Systems-Man-and-Cybernetics-Part-C-Applications-and-Reviews-1094-6977) *31(2):216 - 233 )* | This paper serves as a guide and update for readers working in the CR area. First, the historical evolution of CR systems is presented. |
| **7.** | **Optical Character Recognition by Open-source OCR Tool Tesseract: A Case Study**  *(International Journal Of Computer Applications- December 2022 Edition)* | This paper gives comparative study on commercial OCR tool Transym OCR by considering vehicle number plate as input. |
| **8.** | **A Survey on various Optical Character Recognition Techniques**  *(From IEEE -* [*2018 Conference on Emerging Devices and Smart Systems (ICEDSS)*](https://ieeexplore.ieee.org/xpl/conhome/8520539/proceeding)*)* | Optical Character Recognition consists of various stages includes pre-processing, Classification, Post Acquisition, Pre-Level processing, Segmented Processing, Post-Level processing, Feature Extraction. |

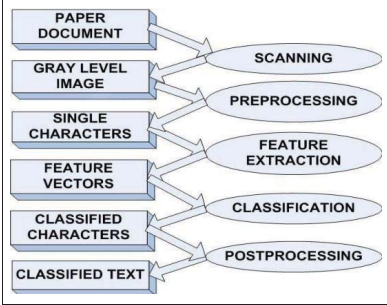
*Table 2.9.1: Table of Literature Survey*

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 SYSTEM MODEL:**

Following is the Figure 1, which shows the classic workflow model of OCR system that follows nine steps (excluding first and the last step) to extract the text from the document. These steps can be classified to main five steps that are, preprocessing, segmentation, feature extraction, classification and recognition.



**Fig: 3.1.1** - **CLASSIC OCR PROCEDURE MODEL**

* + 1. **APPLICATIONS OF OCR:**

OCR Applications has been performed in a numerous of applications. We discussed some of these application areas in this section.

1. **Handwriting Recognition:**

Handwriting recognition is the capacity of a PC to get and translate intelligible handwritten data from sources, for example, paper records, photos, touch-screens and different gadgets.

1. **Receipt Imaging:**

In government offices and autonomous organizations, OCR simplifies information gathering and analysis, among different procedures.

1. **Banking:**

Another imperative use of OCR is in banking, where it is utilized to process cheques without human intervention. A cheque can be embedded with a machine where the framework filters the sum to be issued and the right measure of cash is exchanged. This innovation has been idealized for printed cheque, and is genuinely precise for handwritten checks diminishing the hold-up time in banks.

1. **Automatic Number Plate Recognition:**

Automatic number plate recognition is utilized as a mass observation method making utilization of optical character recognition on pictures to recognize vehicle registration plates. ANPR has additionally been made to store the pictures caught by the cameras including the numbers caught from license plate.

1. **Automatic data entry for documents such as cheques, invoices, and receipts.**
2. **Passport identity verification at the airports.**
3. **Automatic customer insurance claim registration.**
4. **Traffic lights sign recognition.**
5. **Create digital text version of the printed documents such as old paper books, bank database repository.**
6. **Create electronic version, which can be searched and traversed, for the printed documents, LIKE PDF, XML, HTML.**
7. **Technology to read-out the text for helping the visually impaired or blind people.**

**3.2 FUNCTIONAL REQUIREMENTS:**

We have classified these functional requirements as follow:

1. Taking/ choosing the desired text image.

 2. Recognition of the text.

3. Copying the text for different uses

* **Taking/ choosing the desired text image:**

The most important thing here is the use of a PC. The user can input a picture of a text image or choose one from the file directory.

* **Recognition of the text: Description:** The text will be recognized from the image inputted by the user or from any chosen image from the system file directory. The text will be recognized and ready to be use:

1. Recognition of the text from the image.
2. Ready to be used.

* **Copying the text for different uses:** Description: Once the text is recognized and ready to be used, the user will be able to copy, edit, and modify it. Copy the text from the text from the image and modify it.

**3.2.1 MODULES AND THEIR FUNCTIONALITIES**

Our project Optical Character Recognition can be divided into three modules based on its functionality.

The modules classified are as follows: -

1. Document Processing Module
2. System Training Module.
3. Document Recognition Module.

* **DOCUMENT PROCESSING MODULE**

This module performs certain activities such as scanning documents, storing them as images, recognizing characters in images to transfer them into word format. During the recognition process, this module uses the OCR methodology. The module supports the following services: - Scanning printed documents.  Storing the documents as snapshots or images.  Processing those image-based documents.  Recognizing the characters in documents.

* **SYSTEM TRAINING MODULE**

Before converting the printed documents in to editable and searchable documents, the first and the mandatory step is providing training to the system. Here training in the sense, the font followed in the scanned document should be identified by the user. Then the user types all the characters that are required for recognition from the scanned document as an image file. This image file should be provided as an input during the training process. The system gets familiar with the new font. This module supports: - Training the system with the pre-defined fonts.  Training the system with the new fonts that are not present in the system and that cannot be identified by the system.

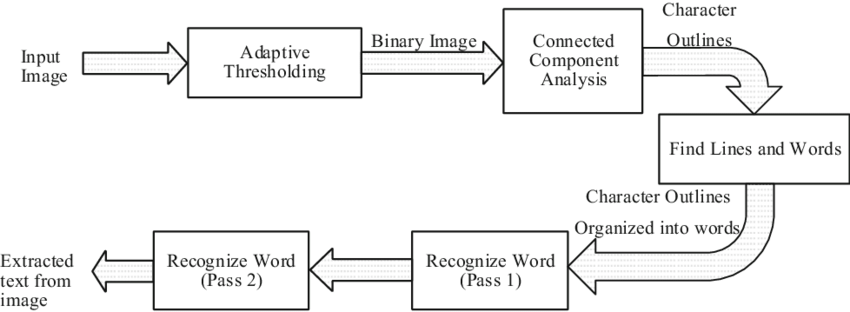
* **DOCUMENT RECOGNITION MODULE**

Once the printed documents are converted into structured documents, any user can recognize the characters present in the document. That means the user can recognize the characters of any language he chooses which makes OCR more flexible. This flexibility is due to the adaptation of grid infrastructure. This is the module where the main functionality of OCR is tested. Under this module, there are two types of recognition. They are handwritten recognition and scanned document recognition.

**3.3 SYSTEM ARCHITECTURE:**

The Architecture of the optical character recognition system consists of the three main components. They are: -

1. Processing
2. OCR
3. Output

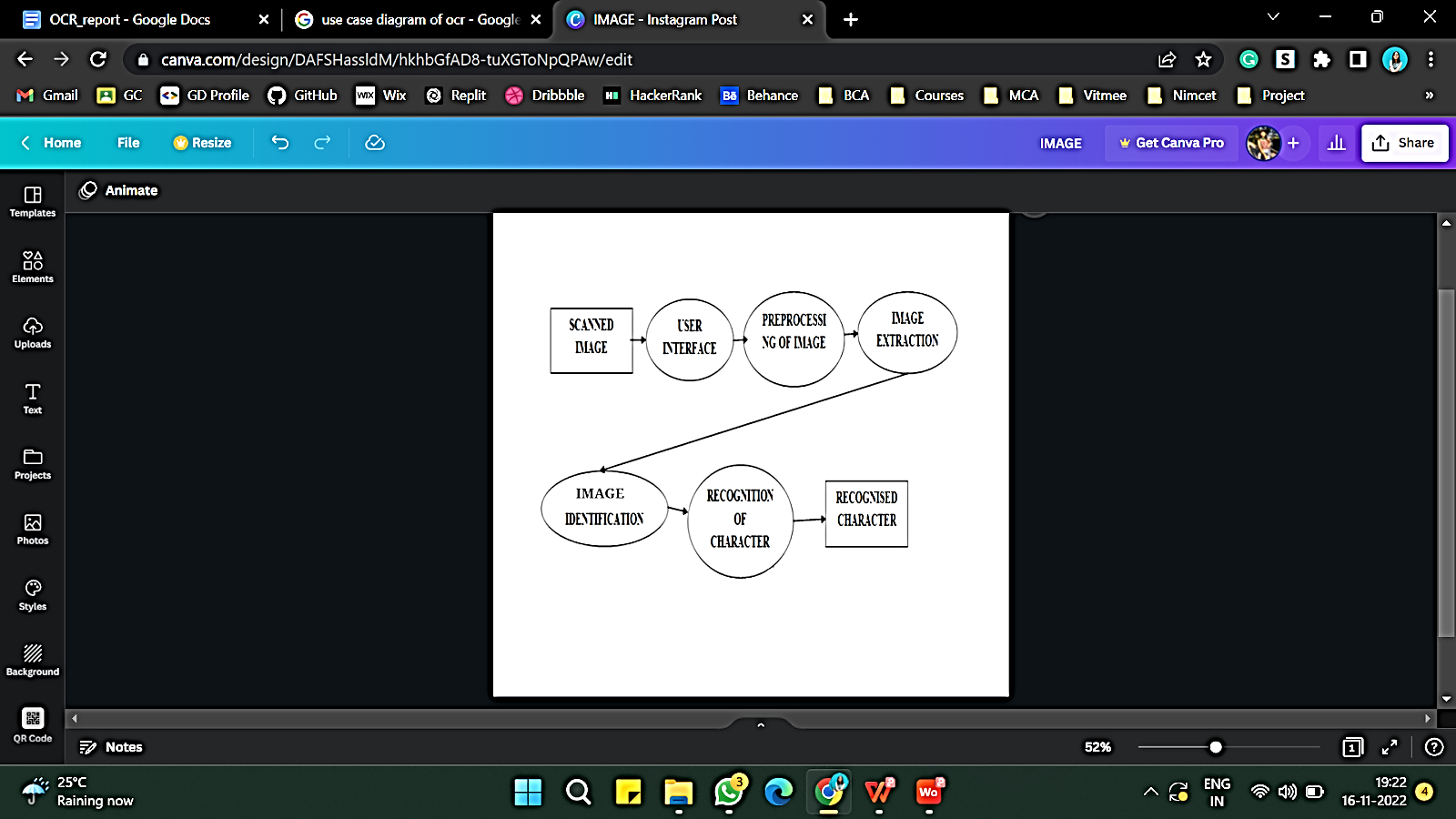
****

**FIG 3.3.1 – ARCHITECTURE OF OCR SYSTEM**

* 1. **UML DIAGRAMS:**
     1. **DATA FLOW DIAGRAM:**

The DFD is also called as bubble chart. A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFD’s can also be used for the visualization of data processing. The flow of data in our system can be described in the form of dataflow diagram as follows: -

1. Firstly, if the user inputs a scanned image in the system.
2. Then, the preprocessing of the scanned image is done.
3. After the preprocessing, the image extraction and detection is done.
4. Lastly, the characters are recognized and the output is generated.

****

**FIG 3.4.1.1- DATA FLOW DIAGRAM OF OCR**

* + 1. **USE CASE DIAGRAM:**

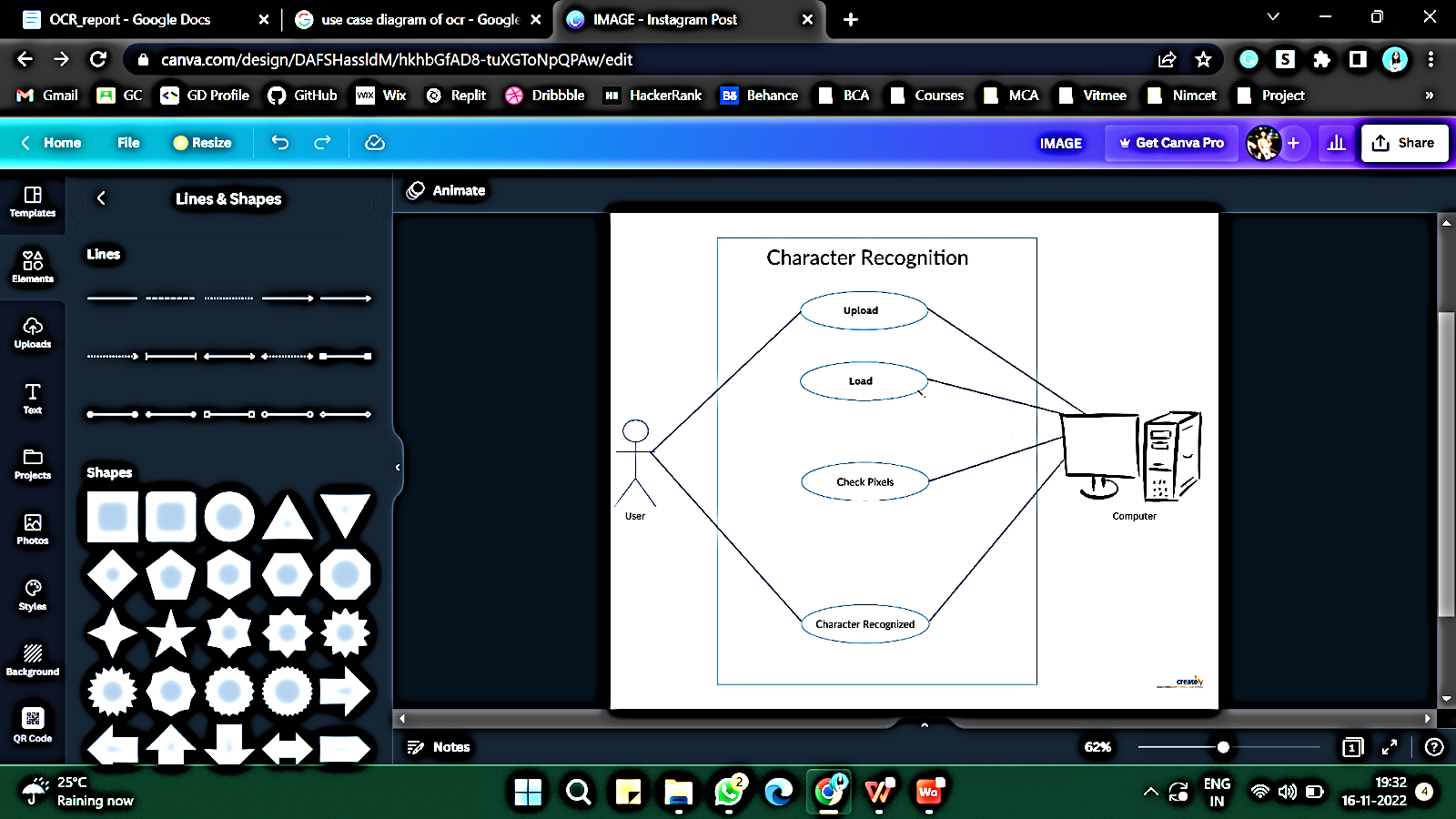
Here in this UML diagram, the image extraction is done in following steps:

1. ACTOR:

* User
* computer

1. RELATIONSHIPS:

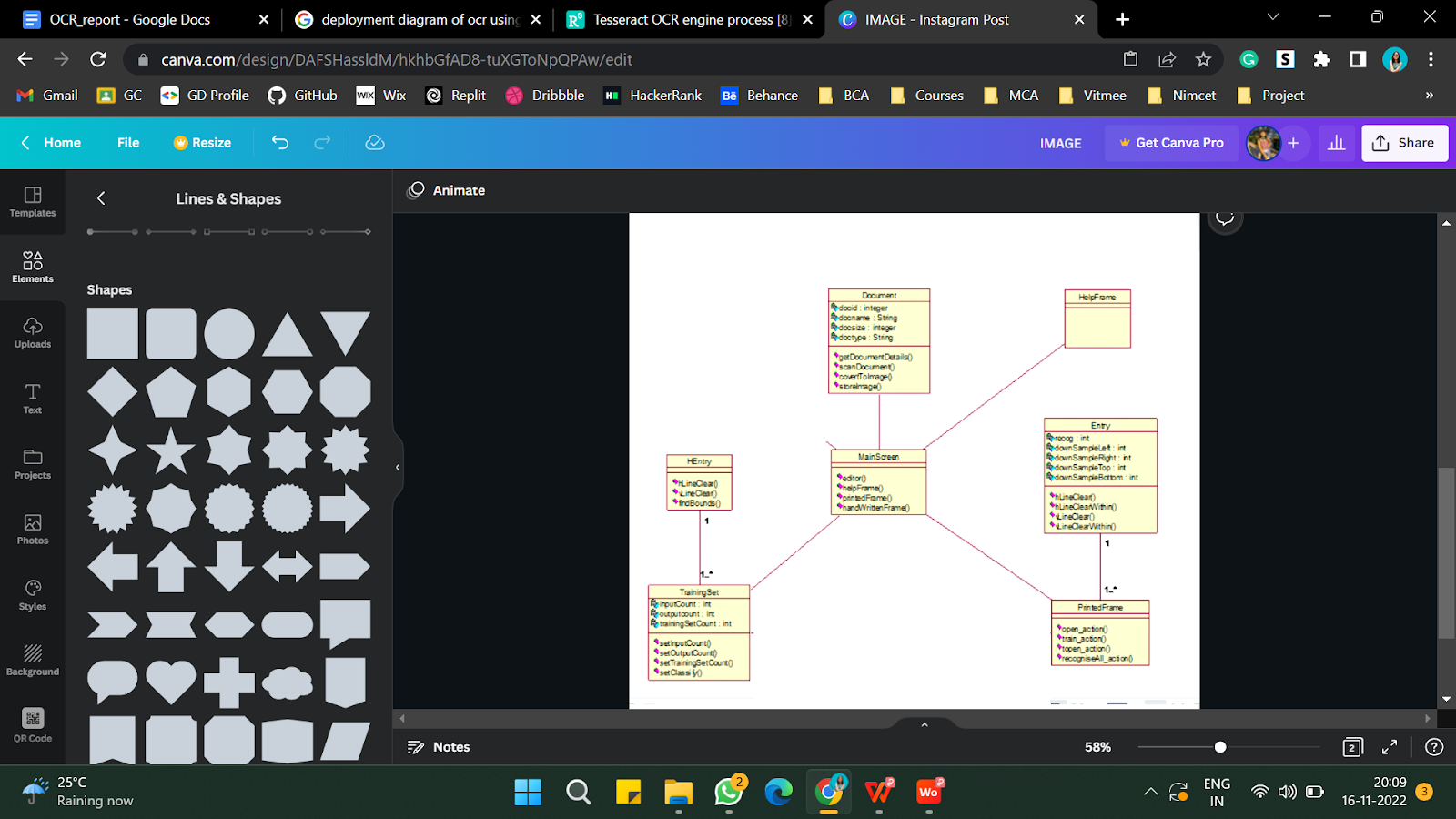
* Uploading of image
* Output generation – Character recognition



**FIG 3.4.2.1 - UML DIAGRAM OF OCR**

**3.4.3 CLASS DIAGRAM:**

The class diagram is the main building block in object-oriented modeling. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed.



**FIG 3.4.3.1- CLASS DIGRAM OF OCR**

* + 1. **SEQUENCE DIAGRAM:**

Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams. Sequence Diagram for Document Processing:

 1. Objects

1. User – ‘U’
2. Computer – ‘C’

 2. Links

1. User to computer

2. User to computer

3. Training to user

4. User to testing

5. Computer to output

3. Messages

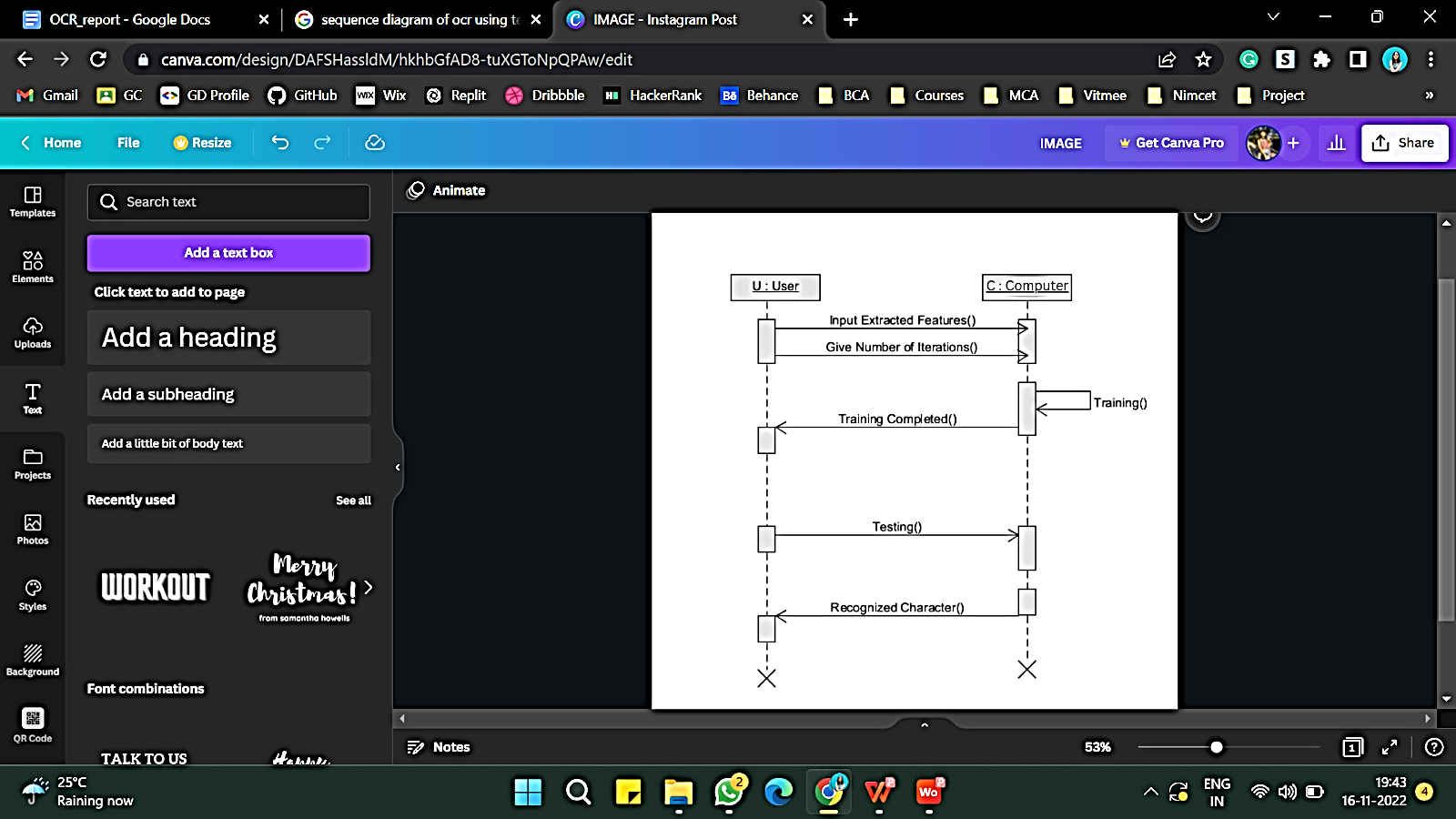
1. Input extracted features

2. Give number of iterations

3. Training completed

4. Testing

5. Recognized character

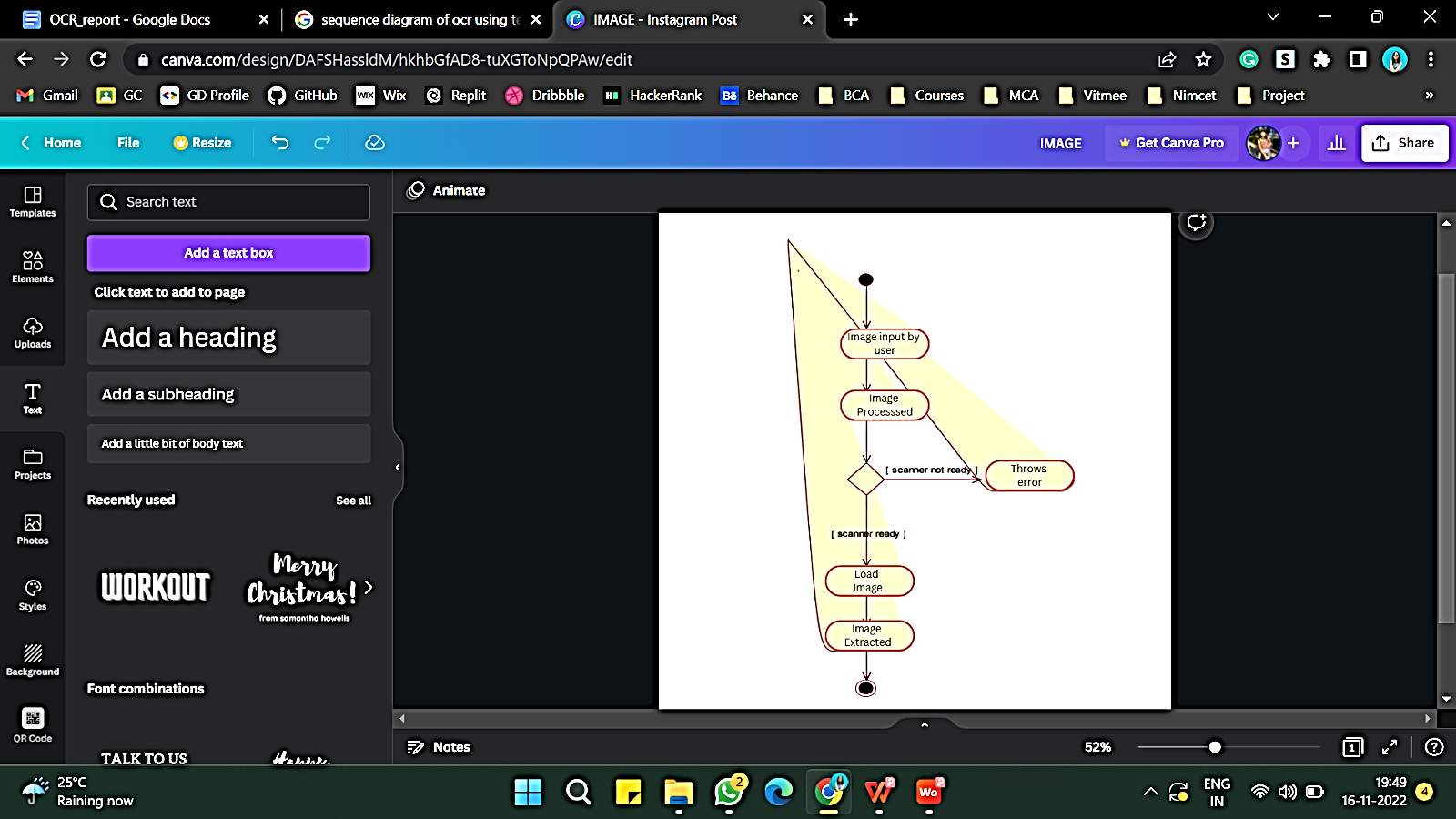


**FIG 3.4.4.1- SEQUENCE DIAGRAM FOR PROCESSING OF OCR**

* + 1. **ACTIVITY DIAGRAM:**

Activity diagrams are probably the most important UML diagrams for doing [business process modelling](https://tallyfy.com/business-process-modeling/). In software development, it is generally used to describe the flow of different activities and actions. These can be both sequential and in parallel.

It contains author, editor and publisher.



**FIG 3.4.5.1- ACTIVITY DIAGRAM FOR PROCESSING OF OCR**

* + 1. **COMPONENT DIAGRAM:**

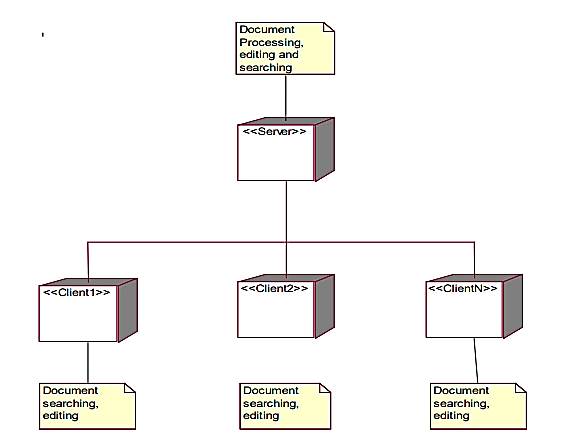
The crucial component in our component diagram that plays a major role in implementing the OCR system is the processing component.  All other components that is Document processing and recognition, Document editing and Document Searching depends on it. They are as follows: -



**FIG 3.4.6.1- COMPONENT DIAGRAM OF OCR**

* + 1. **DEPLOYMENT DIAGRAM:**

 A deployment diagram serves to model the physical deployment of artifacts on deployment targets.



**FIG 3.4.7.1- DEPLOYMENT DIAGRAM**

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 TOOLS AND ENVIRONMENT**

For this Optical character recognition system, we have used following frameworks / programming languages:

1. PROGRAMMING LANGUAGE
   1. PYTHON
2. IDE AND SERVER
   1. JUPYTER

* 1. **DOMAIN AND LANGUAGES:**

1. **DOMAIN: MACHINE LEARNING**

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

Machine learning OCR or deep learning OCR is a group of computer vision problems in which written text from digital images is processed into machine readable text.

1. **PYTHON:**

Python OCR is a technology that recognizes and pulls out text in images like scanned documents and photos using Python. It can be completed using the open-source OCR engine Tesseract. We can do this in [Python](https://builtin.com/learn/tech-dictionary/python) using a few lines of code. One of the most common OCR tools that are used is the [Tesseract](https://github.com/tesseract-ocr/tesseract). Tesseract is an optical character recognition engine for various operating systems.

**PYTHON: Following Python libraries are used in our project:**

**a) OpenCV:** OpenCV is a library for different programming functions primarily that aimed to provide real-time computer vision. It is used to import and perform segmentation of the image as well as extract the image in our project.

**b) Pytesseract:** Pytesseract or Python-tesseract is an OCR tool for python that also serves as a wrapper for the Tesseract-OCR Engine. It can read and recognize text in images and is commonly used in python ocr image to text use cases. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others.

**c) Tesseract:** Tesseract is a library, which provides Optical Character Recognition engine with support for the Unicode and has the ability to recognize more than 100 languages in-built. It can also be trained to recognize other languages as well.

**d) Import OS:** Python OS module provides the facility to establish the interaction between the user and the operating system. The OS comes under Python's standard utility modules. This module offers a portable way of using operating system dependent functionality.

**e) PIL:** Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating and saving images.  Pillow supports a large number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.  
This module is not preloaded with Python. So, to install it execute the following command in the command-line: pip install pillow.

**f) GTT S**: **GTTS** (Google Text-to-Speech), a Python library and CLI tool to interface with Google Translates text-to-speech API. Write spoken mp3 data to a file, a file-like object (byte string) for further audio manipulation, or stdout. Or simply pre-generate Google Translate TTS request URLs to feed to an external program.

**g) PyPDF2:** PyPDF2 is a free and open-source pure-python PDF library capable of splitting, [merging](https://pypdf2.readthedocs.io/en/latest/user/merging-pdfs.html), [cropping, and transforming](https://pypdf2.readthedocs.io/en/latest/user/cropping-and-transforming.html) the pages of PDF files. It can also add custom data, viewing options, and [passwords](https://pypdf2.readthedocs.io/en/latest/user/encryption-decryption.html) to PDF files. PyPDF2 can [retrieve text](https://pypdf2.readthedocs.io/en/latest/user/extract-text.html) and [metadata](https://pypdf2.readthedocs.io/en/latest/user/metadata.html) from PDFs as well.

**4.3 IDE AND SERVER**

1. **IDE:**

**JUPYTER:**

Jupyter Notebooks are a community standard for communicating and performing interactive computing. They are a document that blends computations, output, explanatory text, mathematics, images, and rich media representations of objects.

JupyterLab is one interface used to create and interact with Jupyter Notebook

1. **SERVER:**

* The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.
* Jupyter Server is the backend—the core services, APIs, and [REST endpoints](https://petstore.swagger.io/?url=https://raw.githubusercontent.com/jupyter/jupyter_server/main/jupyter_server/services/api/api.yaml)—to Jupyter web applications.
* Jupyter Server is a replacement for the Tornado Web Server in [Jupyter Notebook](https://github.com/jupyter/notebook). Jupyter web applications should move to using Jupyter Server. For help, see the [Migrating from Notebook Server](https://jupyter-server.readthedocs.io/en/latest/operators/migrate-from-nbserver.html#migrate-from-notebook) page.

**4.4 SOFTWARE REQUIREMENT SPECIFICATION:**

* Operating System: Windows 10/11
* Programming Language: Java
* IDE: Jupyter Notebook (Anaconda PowerShell)

**4.5 HARDWARE REQUIREMENTS SPECIFICATION HARDWARE**

* Processor: AMD Ryzen 5 5500U with Radeon Graphics with 2.10 GHzr or any other
* RAM: Minimum of 512 MB RAM
* Memory:500 MB or higher

**CHAPTER 5**

**TESTING**

**5.1** **TESTING:**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

**5.2 TEST CASES:**

**5.2.1 Importing Modules**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST CASE NO.** | **TEST CASE NAME** | **TEST DATA** | **EXPECTED O/P** | **RESULT** |
| **1** | **OpenCV** | import cv2 | It should be imported | Imported successfully. |
| **2** | **Pytesseract** | import pytesseract as pt | It should be imported | Imported successfully. |
| **3** | **GTTS** | from gtts import gTTS | It should be imported | Imported successfully. |
| **4** | **PyPDF2** | import PyPDF2 | It should be imported | Imported successfully. |

**TABLE 5.2.1.1-IMPORTING MODULES IN OCR**

**5.2.2 Uploading files from directory**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST CASE NO.** | **TEST CASE NAME** | **TEST DATA** | **EXPECTED O/P** | **RESULT** |
| **1** | **jpg** | test\_image\_files = os.listdir(test\_img\_path) | It should upload all jpg images from directory. | Uploaded successfully. |
| **2** | **png** | test\_image\_files = os.listdir(test\_img\_path) | It should upload all png images from directory. | Uploaded successfully. |
| **3** | **pdf** | test\_image\_files = os.listdir(test\_img\_path) | It should upload all pdf files from directory. | Uploaded successfully. |
| **4** | **Text file** | test\_image\_files = os.listdir(test\_img\_path) | It should upload all text files from directory. | Uploaded successfully. |

**TABLE 5.2.2.1-UPLOADING FILES**

**5.2.3 Uploading languages**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST CASE NO.** | **TEST CASE NAME** | **TEST DATA** | **EXPECTED O/P** | **RESULT** |
| **1** | **English** | bound-text-1.jpg |  |  |
| **2** | **Hindi** | hindi-text-1.jpg |  |  |
| **3** | **Portuguese** | portu-text-1.jpg |  |  |
| **4** | **Sin** | sin-text-2.gif |  |  |
| **5** | **Japanese** | jap-text-2.png |  |  |
| **6** | **Tamil** | tam-text-1.png |  |  |

**TABLE 5.2.3.1-UPLOADING LANGUAGES**

**5.2.4 Extract text from various sources**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST CASE NO.** | **TEST CASE NAME** | **TEST DATA** | **EXPECTED O/P** | **RESULT** |
| **1** | **jpg** | bound-text-1.jpg |  |  |
| **2** | **png** | news-1.png |  |  |
| **3** | **pdf** | report\_ijamec.pdf |  |  |
| **4** | **folder** | bound-text-1& bound-text-2 from folder |  |  |
| **5** | **text** | image-paths.txt |  | No output |

**TABLE 5.2.4.1-EXTRACTING TEXT FROM VARIOUS SOURCES**

**5.2.5 Extract text from an image with timeout**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST CASE NO.** | **TEST CASE NAME** | **TEST DATA** | **EXPECTED O/P** | **RESULT** |
| **1** | **Timeout Extraction** | News-2.jpg with 0.2ms |  |  |
| **2** | **Timeout Extraction** | News-2.jpg with 2ms |  |  |

**TABLE 5.2.5.1-TIMEOUT EXTRACTION**

**5.2.6 Converting image into various file formats**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST CASE NO.** | **TEST CASE NAME** | **TEST DATA** | **EXPECTED O/P** | **RESULT** |
| **1** | **Converting into pdf** | news-1.png |  |  |
| **2** | **Converting into xml** | news-1.png |  |  |
| **3** | **Converting into HOCR** | news-1.png |  |  |
| **4** | **Converting into audio** | letter-1.png |  |  |

**TABLE 5.2.6.1-CONVERTING IMAGES INTO OTHER FORMATS**

**CHAPTER 6**

**RESULTS**

**6.1 IMPORTING MODULE**

import os

import cv2

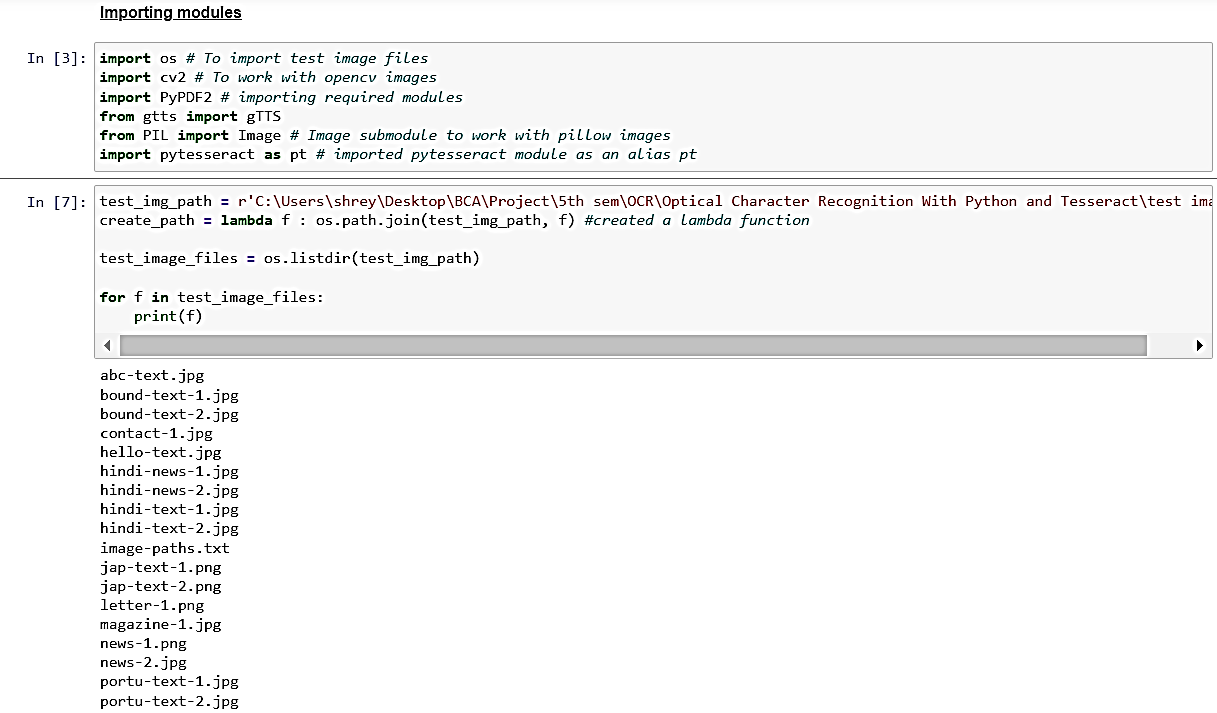
import PyPDF2

from gtts import gTTS

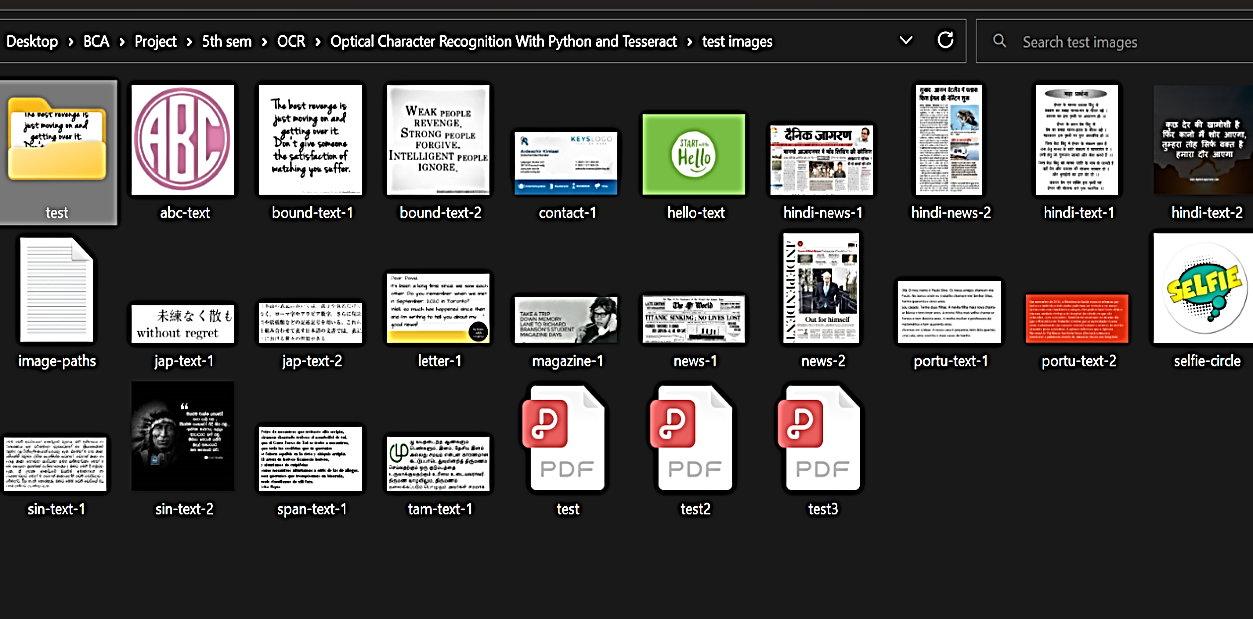
from PIL import Image

import pytesseract as pt

**6.2 IMPORTING ALL IMAGES FROM PATH**

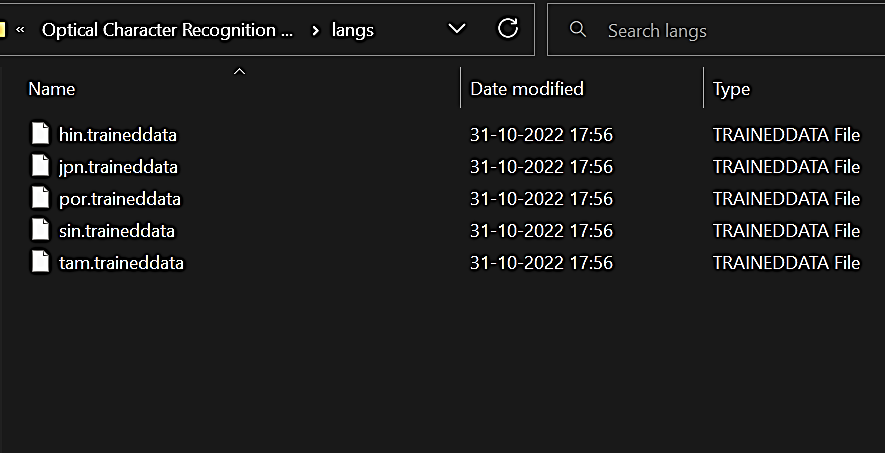


**FIG 6.2.1- IMPORTING IMAGES FROM THE PATH**



**FIG 6.2.2- IMAGES IN THE PATH**

**6.3 IMPORTING THE DOWNLOADED LANGUAGES**



**FIG 6.3.1- IMPORTING LANGUAGES FROM FILE**

**6.4 EXTRACT TEXT FROM AN IMAGE: SIMPLE**

image\_path = test\_image\_files[1] # 2, 3, 12, 1, 13, 15

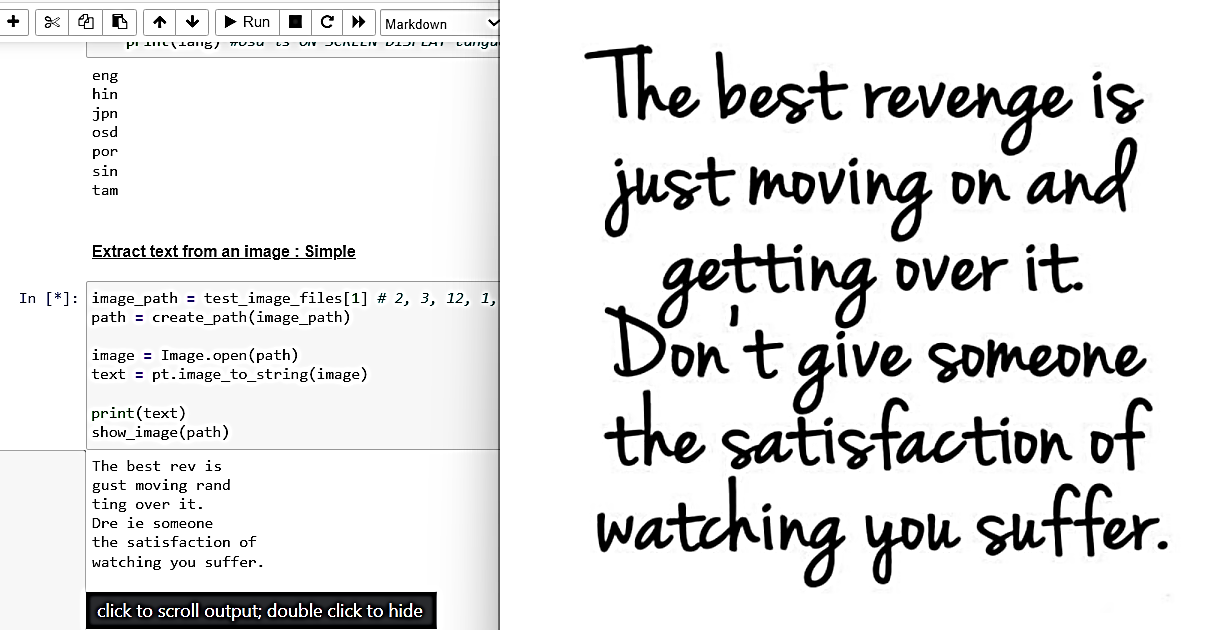
path = create\_path(image\_path)

image = Image.open(path)

text = pt.image\_to\_string(image)

print(text)

show\_image(path)



**FIG 6.4.1- SIMPLE IMAGE TEXT EXTRACTION**

**6.5 EXTRACT TEXT FROM AN IMAGE: SPECIFYING A LANGUAGE**

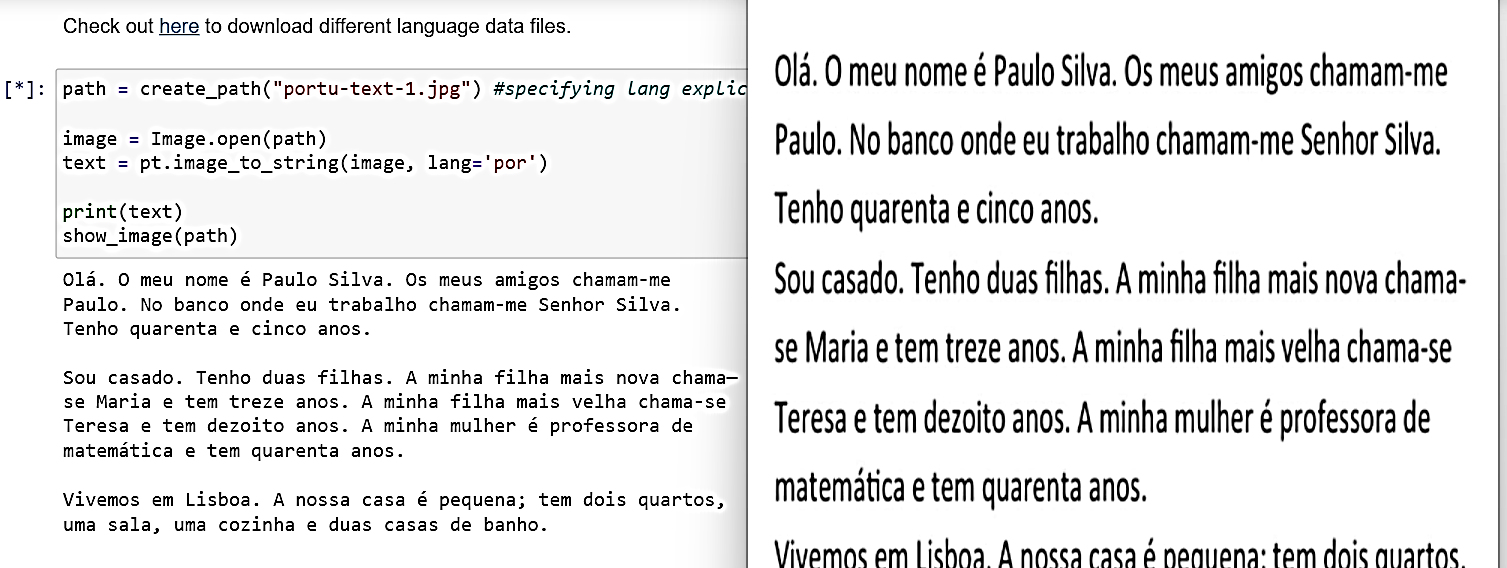
path = create\_path("portu-text-1.jpg")

image = Image.open(path)

text = pt.image\_to\_string(image, lang='por')

print(text)

show\_image(path)



**FIG 6.5.1- TEXT EXTRACTION OF SPECIFIC LANGUAGE**

**6.6 EXTRACT TEXT FROM AN IMAGE: TIMEOUT EXTRACTION**

path = create\_path("news-2.jpg")

image = Image.open(path)

text = 'NO TEXT TO BE APPEARED'

try: text = pt.image\_to\_string(image, lang='eng', timeout=5) #giving 0.5s to tesseract to extract image

except RuntimeError as timeout\_error:

    print("[TIMEOUT ERROR]")

print(text)

show\_image(path)



**FIG 6.6.1 – TIMEOUT TEXT EXTRACTION**

**6.7 GET BOUNDING BOX ESTIMATES**

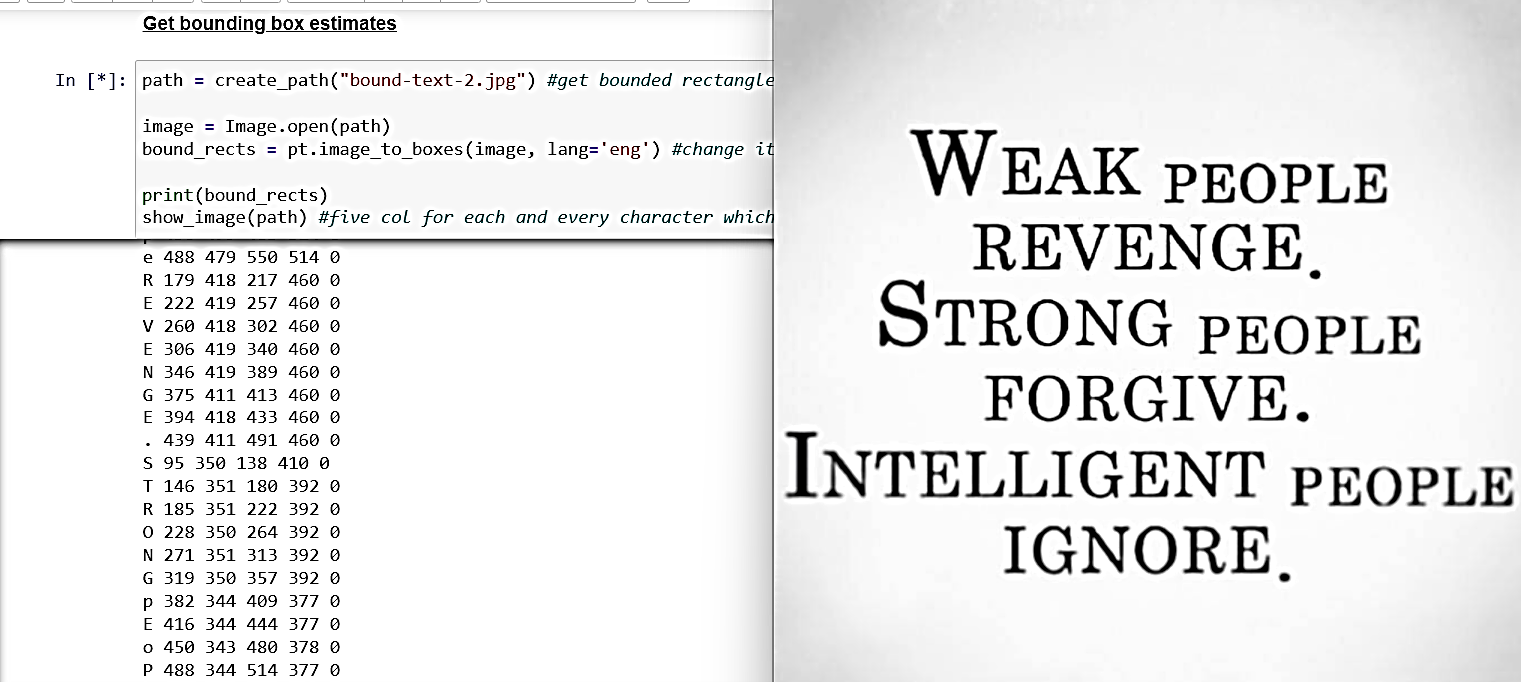
path = create\_path("bound-text-2.jpg")

image = Image.open(path)

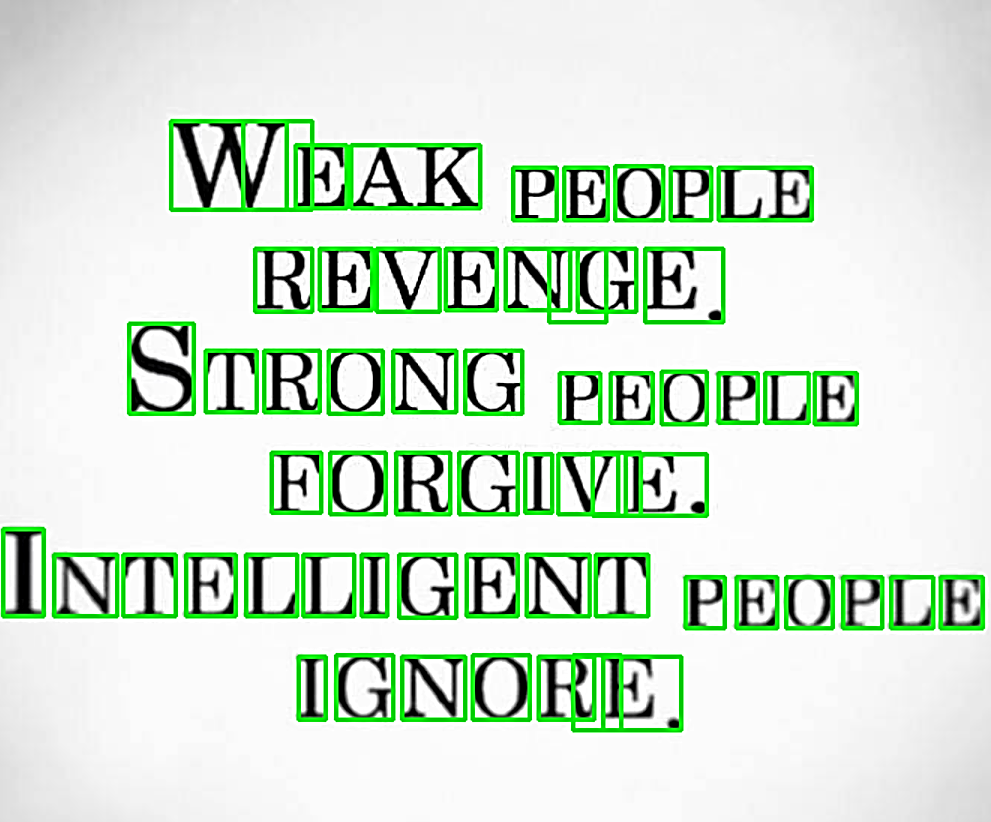
bound\_rects = pt.image\_to\_boxes(image, lang='eng')

print(bound\_rects)

show\_image(path)



**FIG 6.7.1 – BOUNDING BOX ESTIMATES**



**FIG 6.7.2 – BOUNDING BOX OUTPUT**

**6.8 GET VERBOSE DATA INCLUDING BOXES, CONFIDENCES, LINE AND PAGE NUMBERS:**

image\_path = test\_image\_files[3]

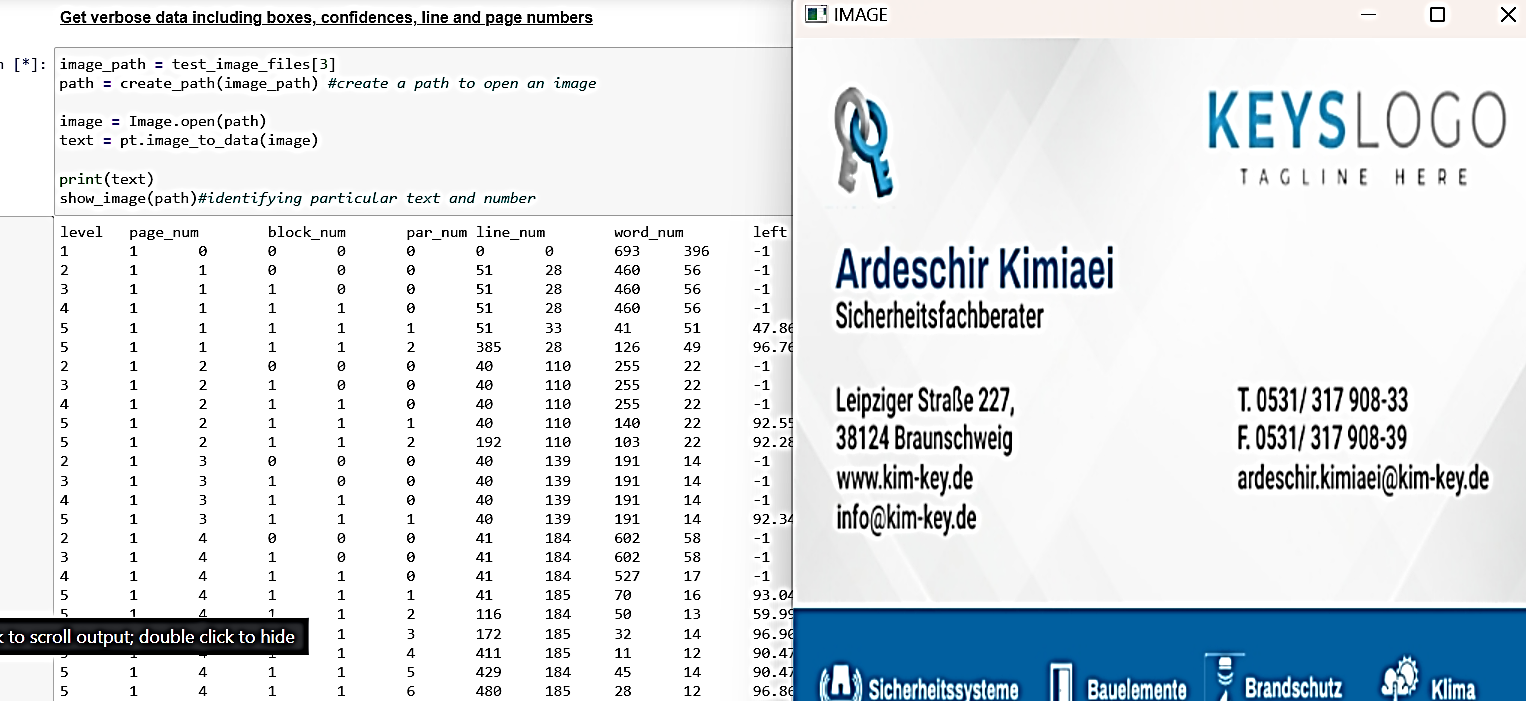
path = create\_path(image\_path)

image = Image.open(path)

text = pt.image\_to\_data(image)

print(text)

show\_image(path)



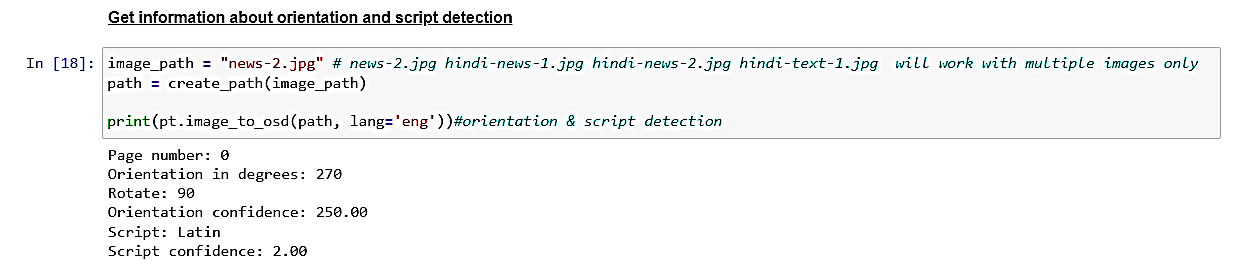
**FIG 6.8.1 – VERBOSE DATA EXTRACTION**

**6.9 GET INFORMATION ABOUT ORIENTATION AND SCRIPT DETECTION**

image\_path = "news-2.jpg"

path = create\_path(image\_path)

print(pt.image\_to\_osd(path, lang='eng'))



**FIG 6.9.1 – ORIENTATION AND SCRIPT DETECTION EXTRACTION**

**6.10 CONVERT IN TO DIFFERENT FILE FORMATS (PDF, XML, HOCR)**

image\_path = "news-1.png"

path = create\_path(image\_path)

file\_save\_path = r'C:\Users\shrey\Desktop\BCA\Project\5th sem\OCR\Optical Character Recognition With Python and Tesseract\files'

pdf = pt.image\_to\_pdf\_or\_hocr(path, extension='pdf')

file = open(os.path.join(file\_save\_path, "pdf-content.pdf"), 'w+b')

file.write(pdf)

file.close()

hocr = pt.image\_to\_pdf\_or\_hocr(path, extension='hocr')

file = open(os.path.join(file\_save\_path, "hocr-content.html"), 'w+b')

file.write(hocr)

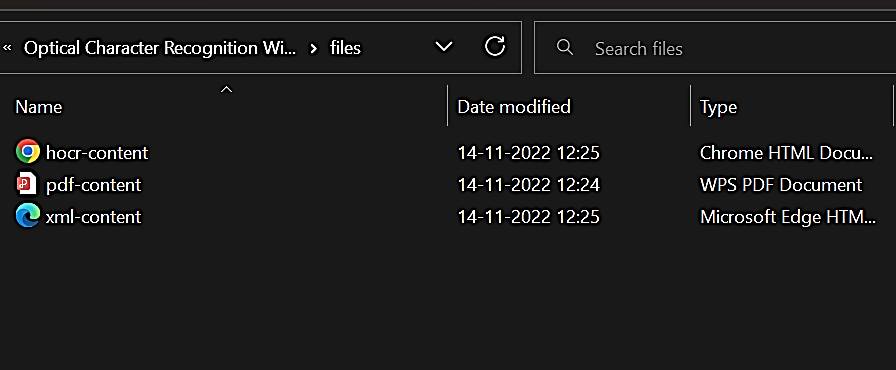
file.close()

xml = pt.image\_to\_alto\_xml(path)

file = open(os.path.join(file\_save\_path, "xml-content.xml"), 'w+b')

file.write(xml)

file.close()



**FIG 6.10.1** - **DIFFERENT FILE FORMATS (PDF, XML, HOCR) EXTRACTION**

**6.11 EXTRACT TEXT FROM PDF:**

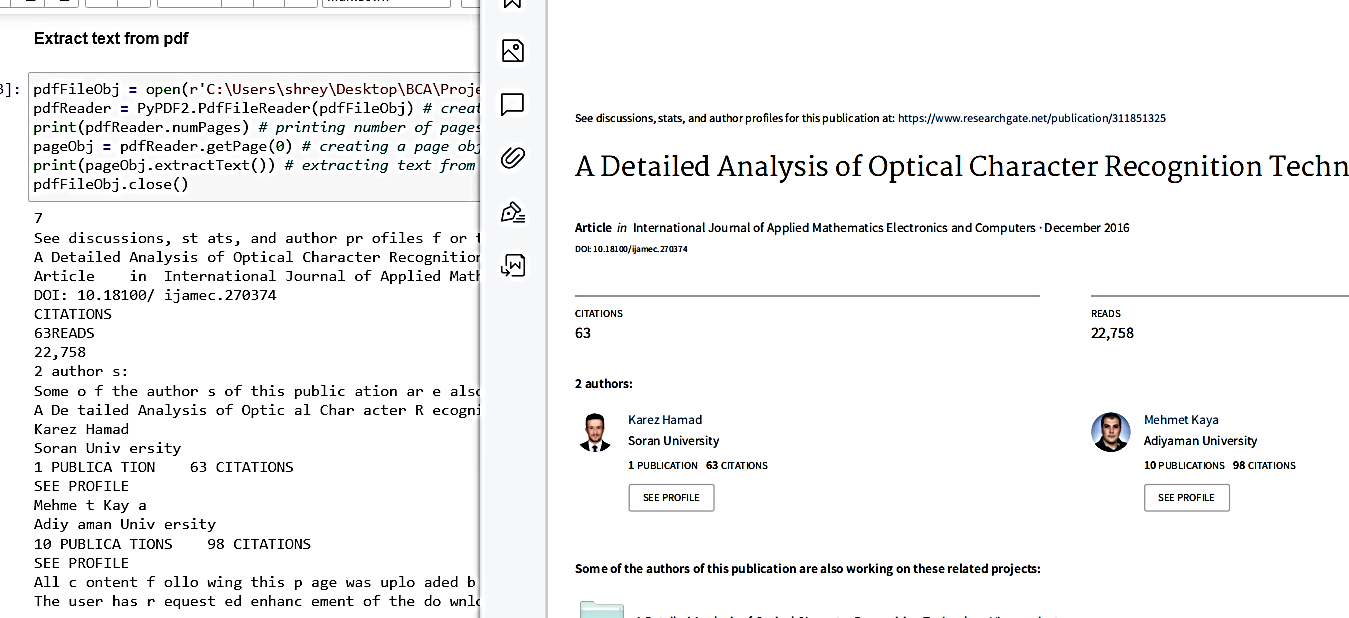
pdfFileObj=open(r'C:\Users\shrey\Desktop\BCA\Project\5th sem\OCR\report\report\_ijamec.pdf','rb')pdfReader= PyPDF2.PdfFileReader(pdfFileObj)

print(pdfReader.numPages)

pageObj = pdfReader.getPage(0)

print(pageObj.extractText())

pdfFileObj.close()



**FIG 6.11.1 – TEXT FROM PDF EXTRACTION**

**6.12 EXTRACT TEXT FROM MULTIPLE IMAGES IN A FOLDER**

path\_to\_images = r'C:/Users/shrey/Desktop/BCA/Project/5th sem/OCR/Optical Character Recognition With Python and Tesseract/test images/test/'

pt.tesseract\_cmd = pt.pytesseract.tesseract\_cmd

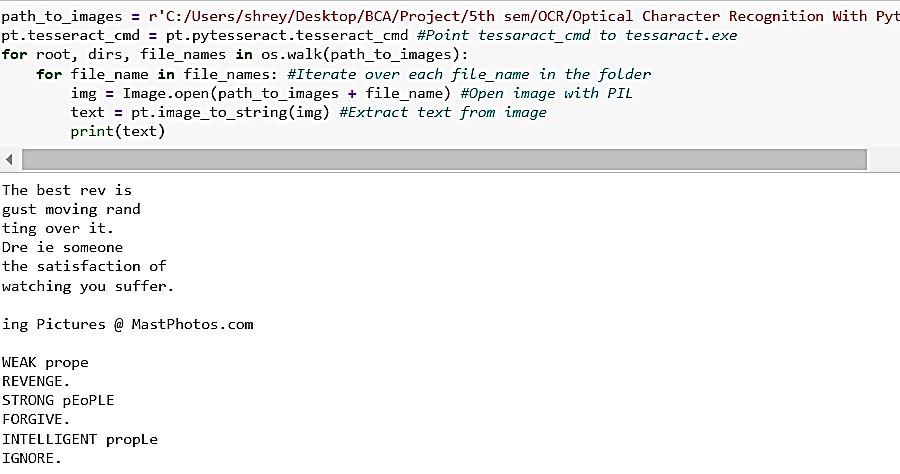
for root, dirs, file\_names in os.walk(path\_to\_images):

    for file\_name in file\_names:

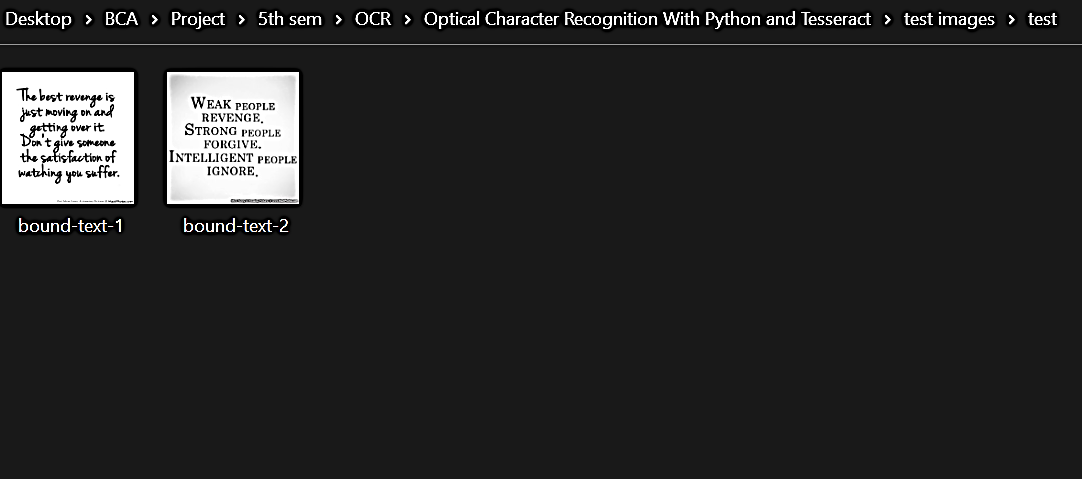
        img = Image.open(path\_to\_images + file\_name)

        text = pt.image\_to\_string(img)

        print(text)



**FIG 6.12.1 - TEXT FROM MULTIPLE IMAGES EXTRACTION**



**FIG 6.12.2 - MULTIPLE IMAGES IN FILE**

**6.13 CONVERTING IMAGE TEXT TO AUDIO**

img = cv2.imread(r'C:\Users\shrey\Desktop\BCA\Project\5th sem\OCR\Optical Character Recognition With Python and Tesseract\test images\bound-text-2.jpg')

img = cv2.resize(img, (600, 360))

hImg, wImg, \_ = img.shape

boxes = pt.image\_to\_boxes(img)

xy = pt.image\_to\_string(img)

for b in boxes.splitlines():

  b = b.split(' ')

x, y, w, h = int(b[1]), int(b[2]), int(b[3]), int(b[4])

cv2.rectangle(img, (x, hImg - y), (w, hImg - h), (50, 50, 255), 1)

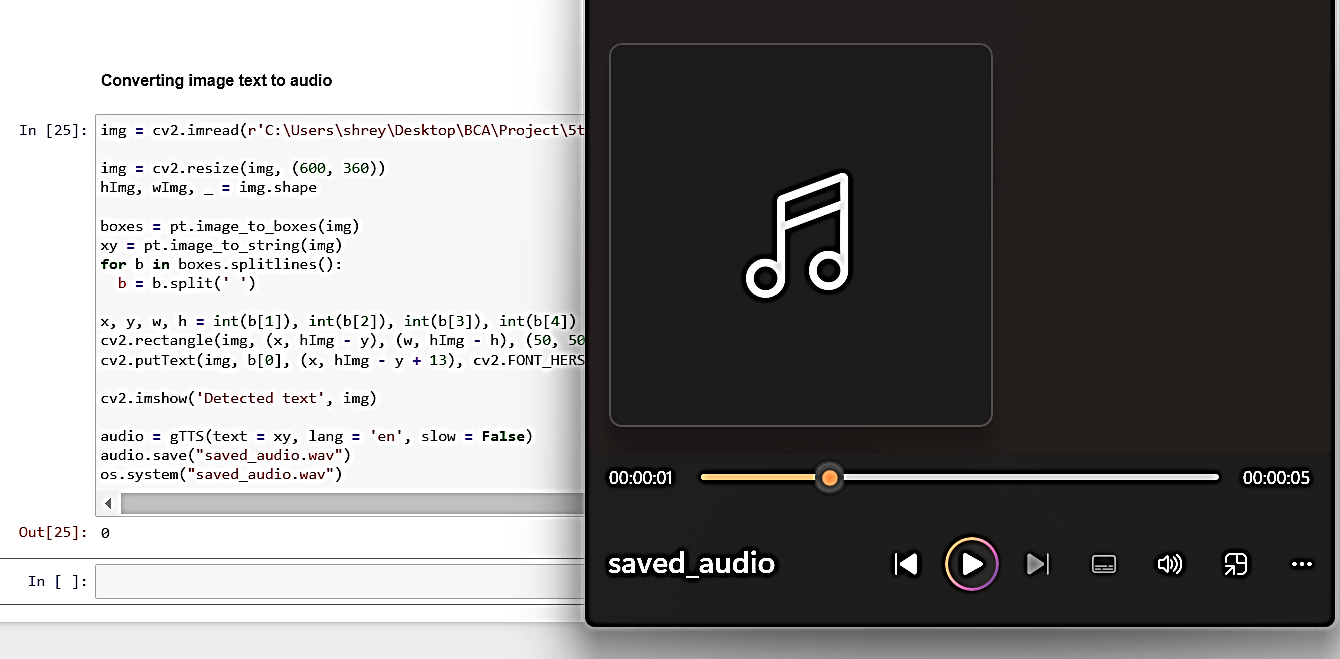
cv2.putText(img, b[0], (x, hImg - y + 13), cv2.FONT\_HERSHEY\_SIMPLEX, 0.4, (50, 205, 50), 1)

cv2.imshow('Detected text', img)

audio = gTTS(text = xy, lang = 'en', slow = False)

audio.save("saved\_audio.wav")

os.system("saved\_audio.wav")



**FIG 6.13.1 – IMAGE TEXT TO AUDIO CONVERSION**

**CHAPTER: 7**

**CONCLUSIONS AND FUTURE ENHANCEMENTS**

**7.1 CONCLUSION**

What does the future hold for OCR? Given enough entrepreneurial designers and sufficient research and development dollars, OCR can become a powerful tool for future data entry applications. However, the limited availability of funds in a capital-short environment could restrict the growth of this technology. But, given the proper impetus and encouragement, a lot of benefits can be provided by the OCR system.

They are: - The automated entry of data by OCR is one of the most attractive, labour reducing technology. The recognition of new font characters by the system is very easy and quick.  We can edit the information of the documents more conveniently and we can reuse the edited information as and when required.  The extension to software other than editing and searching is topic for future works.

**7.2 FUTURE ENHANCEMENTS**

The Optical Character Recognition software can be enhanced in the future in different kinds of ways such as:

* Training and recognition speeds can be increased greater and greater by making it more user-friendly.
* As a future work we are planning to use OCR for such practical applications for daily personal use. We are planning to incorporate mobile devices with OCR in one OCR system.
* UI Development: Better responsive UI with features such as drag-and-drop.
* Tainting Model: Model will be trained with more amount of dataset to output higher accuracy.
* Deploying: The WebApp will be deployed on a host and will be accessible on a domain to people to use as free and contribute-and-learn.

**CHAPTER: 8**

**REFERENCES**

**8.1 REFERENCES:**

Under this references section, we have mentioned various references from which we collected our problem and several others that supported us to design the solution for our problem. These references include either book, papers published through some standards and several websites’ links with URL’s:

1. 2021 International Conference on Emerging Smart Computing and Informatics (ESCI) AISSMS Institute of Information Technology, Pune, India. Mar 5-7, 2021, Optical Character Recognition using Tesseract and Classification

2. International Journal of Applied Mathematics, Electronics and Computers Advanced Technology and Science. A Detailed Analysis of Optical Character Recognition Technology.

3.Compiter Vision and Deep Learning Resource Guide by Dr. Adrian Rosebrock from pyimagesearch.

4. Jaewoo Park; Eunji Lee, Yoonsik Kim, Isaac Kang, Hyung Il Koo and Nam Ik Cho “Multi-Lingual Optical Character Recognition System Using the Reinforcement Learning of Character Segmenter,” in IEEE Access (Volume: 8).

5. Vaibhav. V. Mainkar, Jyoti A. Katkar, Ajinkya B. Upade, Poonam R. Pednekar, “Handwritten Character Recognition to Obtain Editable Text,” 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 599 – 602.

6. N. Arica and F. Yarman-Vural, An Overview of Character Recognition Focused on Offline Handwriting, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 31, No.2, pp. 216--233, 2001.

7. Patel C, Patel A, Patel D. Optical character recognition by open-source OCR tool tesseract: A case study. International Journal of Computer Applications. 2012 Jan 1;55(10).

8. Abin M Sabu and Anto Sahaya Das, “A Survey on various Optical Character Recognition Techniques,” 2018 Conference on Emerging Devices and Smart Systems (ICEDSS)

9.<https://github.com/tesseract-ocr/tesseract>

10. <https://www.thepythoncode.com/article/optical-character-recognition-pytesseract-python>

11.<https://pyimagesearch.com/2021/08/23/your-first-ocr-project-with-tesseract-and-python/>

**----------------------------------------------------------------------------------------------------------**