

# Smart-OCR-for-Document-Digitization

---

## **Index:**

- 1.Introduction.
- 2.Literature survey.
- 3.Theoretical analysis.
- 4.Experimental investigations.
- 5.Flowchart.
- 6.Result.
- 7.Advantages and Disadvantages.
- 8.Applications.
- 9.Conclusion.
- 10.Future scope.
- 11.Bibliography.
- 12.Appendix.

# 1.Introduction

- Humans can understand the contents of an image simply by looking. We perceive the text on the image as text and can read it.
- Computers don't work the same way. They need something more concrete, organized in a way they can understand.
- This is where Optical Character Recognition (OCR) kicks in. Whether it's recognition of car plates from a camera, or hand-written documents that should be converted into a digital copy, this technique is very useful. While it's not always perfect, it's very convenient and makes it a lot easier and faster for some people to do their jobs.
- This project aims at creating an application form where the user can upload a pdf document/Image containing text, the document is analyzed by an Optical character recognition (OCR) to extract text from it. The extracted text is again saved in a text document in the local drive.

## Overview

- 1.Building Text Extraction Python code for Documents.
- 2.Build Flask Application.
- 3.Run and Deploy.

## Purpose

- The main purpose of the OCR service is to convert the images of different types (Printed, Handwritten, etc.) into machine-encoded text (such that the machine can interpret the data easily).

# 2.Literature survey

To complete this project we required the following packages and Softwares:

- Python IDE- For programming(Pycharm).
- pytesseract -OCR package in python
- pdf2image- Converting PDF to Image
- tesseract-ocr execution file -Backend used for pytesseract
- poppler-Supporting file for pdf2image package
- Flask-To Build a web application

## Existing problem

- The main problem with OCR is that it only outputs unstructured characters. This necessitates the combination of other machine learning technologies into OCR. By that, users can reach structured data from their documents.
- Errors include misreading letters, skipping over unreadable letters, or combining text from adjacent columns or image captions. While many factors affect the performance of OCR tools, the number of errors depends on the quality and form of the text, including the font used.
- However, even with high-quality documents, OCR tools can make mistakes because there are a variety of document formats, fonts, and styles for each character.

### Document-based Limitations

- Colored backgrounds: Colorful background patterns can be troublesome because they can decrease text recognition
- Blurry or glared texts: Blurry or glared images are challenging to read for humans as well as computers.
- Skewed or non-oriented documents: For situations where the image may be skewed, OCR will have a harder time identifying the characters because the text is not aligned.

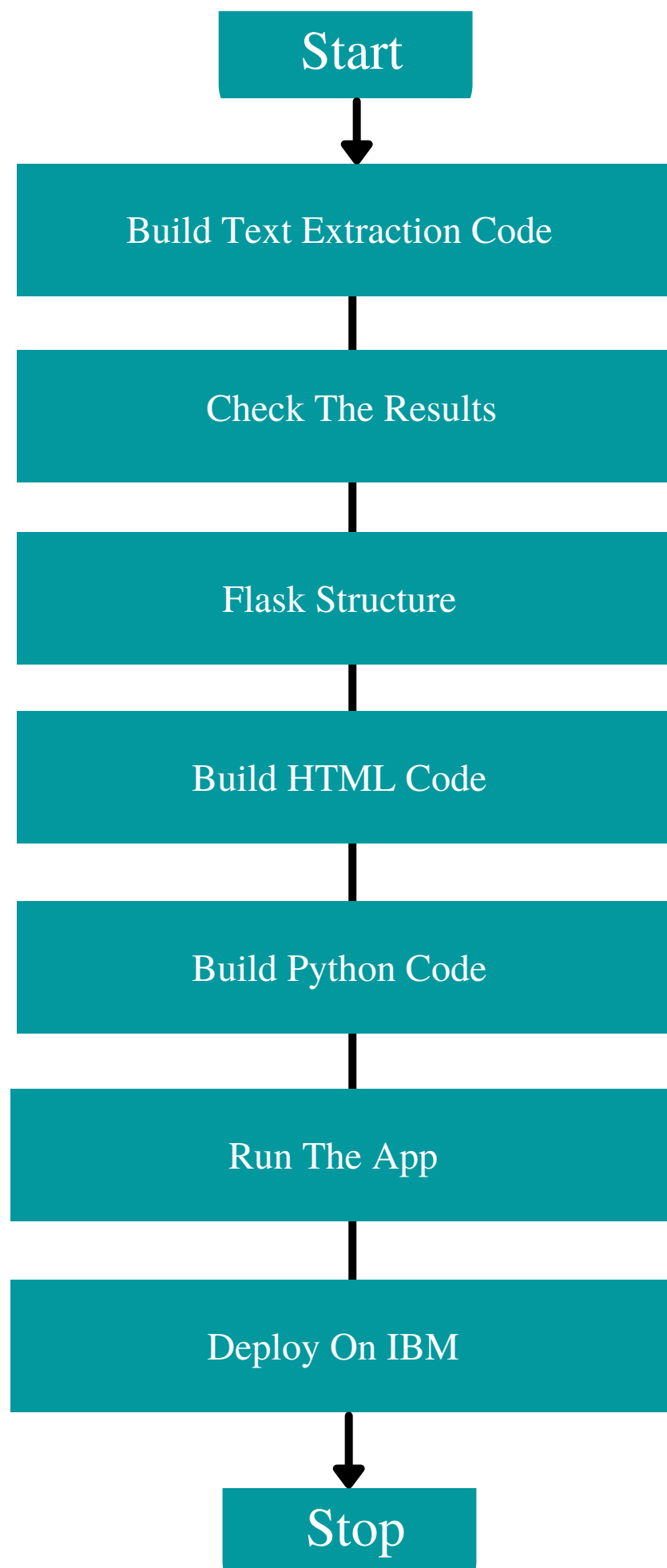
## Proposed Solution

- The image is first scanned and the text and graphics elements are converted into a bitmap, which is essentially a matrix of black and white dots. The image is then pre-processed where the brightness and contrast are adjusted to enhance the accuracy of the process.
- The image is now split into zones identifying the areas of interest such as where the images or text are and this helps kickoff the extraction process. The areas containing text can now be broken down further into lines and words and characters and now the software is able to match the characters through the comparison and various detection algorithms. The final result is the text in the image that we're given.
- The process may not be 100% accurate and might need human intervention to correct some elements that were not scanned correctly. Error correction can also be achieved using a dictionary or even Natural Language Processing (NLP).
- The output can now be converted to other mediums such as word documents and PDFs.

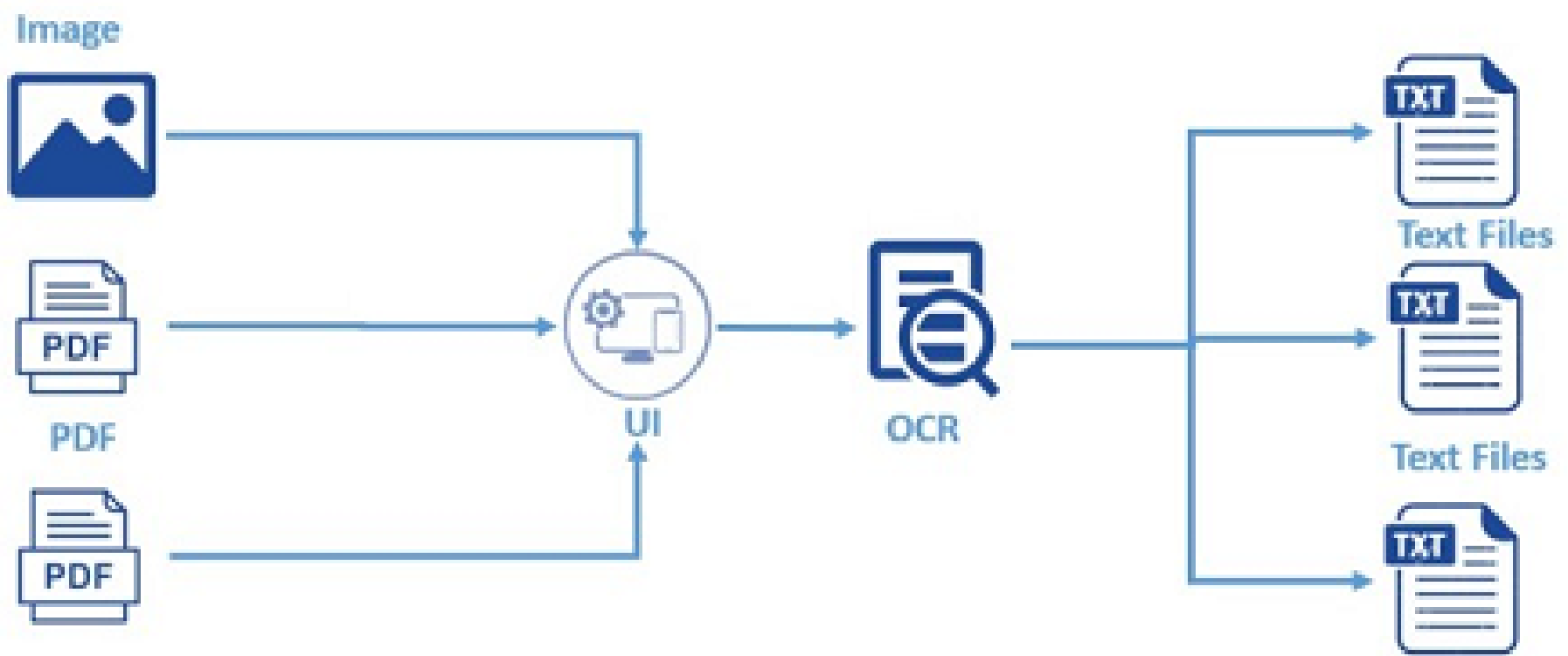
### 3.Theoretical analysis

Block diagram:

Diagrammatic overview of the project



## Architecture overview of the project



## 4. Experimental investigations

During the project we get to know the use-cases and benefits of using the following tools/packages:

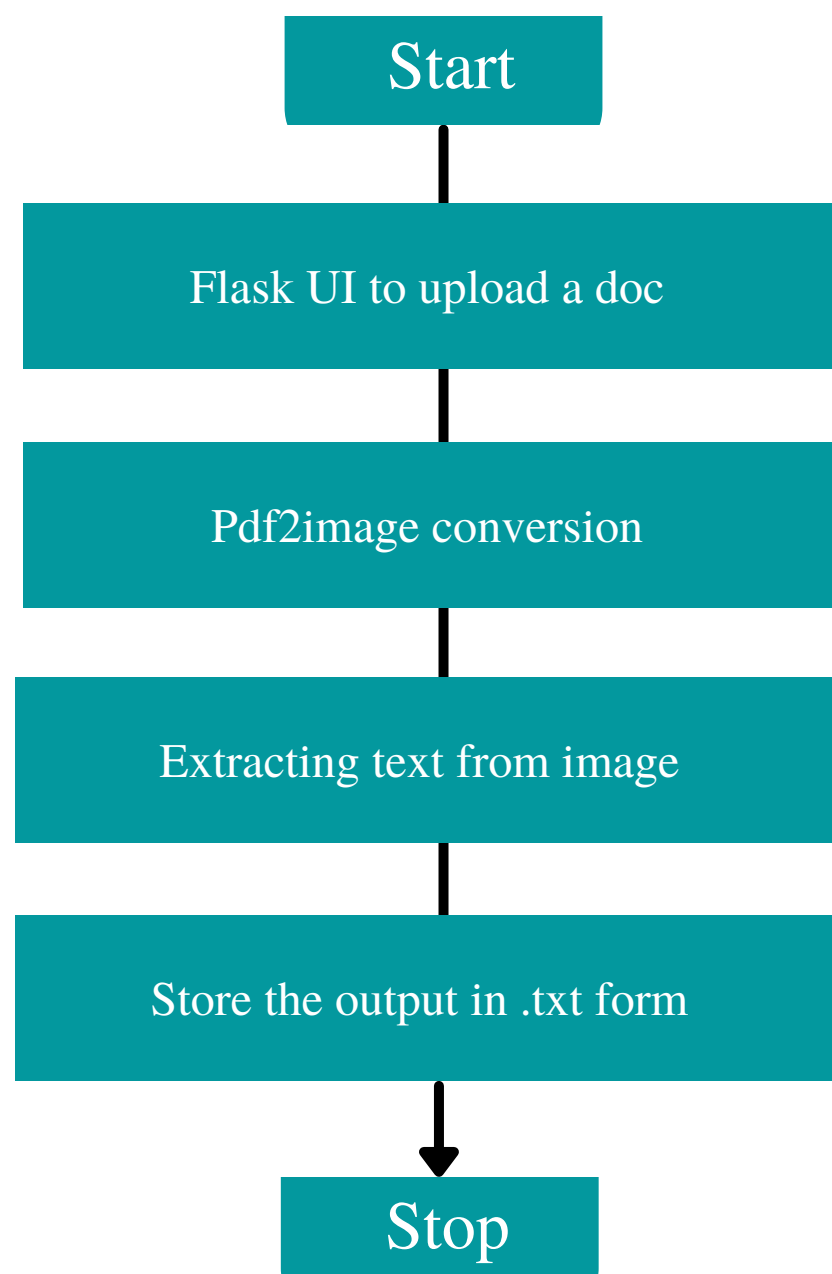
- **Pytesseract:** Pytesseract or Python-tesseract is an Optical Character Recognition (OCR) tool for python. It will read and recognize the text in images, license plates, etc. ... (Any Image with Text). Binarizing the Image (Converting Image to Binary).
- **pdf2image:** A python module that wraps the pdftoppm utility to convert PDF to PIL Image object - Belval/pdf2image.
- **Flask:** Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.
- **Opencv** – It is an Open Source Computer Vision Library which are mainly used for image processing, video capture and analysis including features like face detection.

- For this OCR project, we we used the Python-Tesseract, or simply PyTesseract, library which is a wrapper for Google's Tesseract-OCR Engine.
- We also used the Flask web framework to create our simple OCR server where we can upload photos or pdf's of images for character recognition purposes.
- Besides those, we also used the Pillow library which is a fork of the Python Imaging Library (PIL) to handle the opening and manipulation of images in many formats in Python.

## 5.Flowchart

### Project Flow:

- Upload a pdf document
- Convert PDF document to image
- Extract the text from the image
- Store the extracted text in the text document



## 6.Result

- First we run the app in the local host.
- Now we used a sample pdf file to upload.
- It is converted to a text format and is stored in output folder.
- This could be found as a .txt file with a serial number before it.
- Final findings (output) of the project along with Screenshots.

### A Simple PDF File

This is a small demonstration .pdf file -

just for use in the Virtual Mechanics tutorials. More text. And more text. And more text. And more text.

And more text. And more text. And more text. And more text. And more text. And more text. Boring, zzzzz. And more text. And more text. And more text. And more text. And more text. And more text.

And more text. And more text. And more text. And more text. And more text. And more text. And more text. Even more. Continued on page 2 ...

Smart OCR

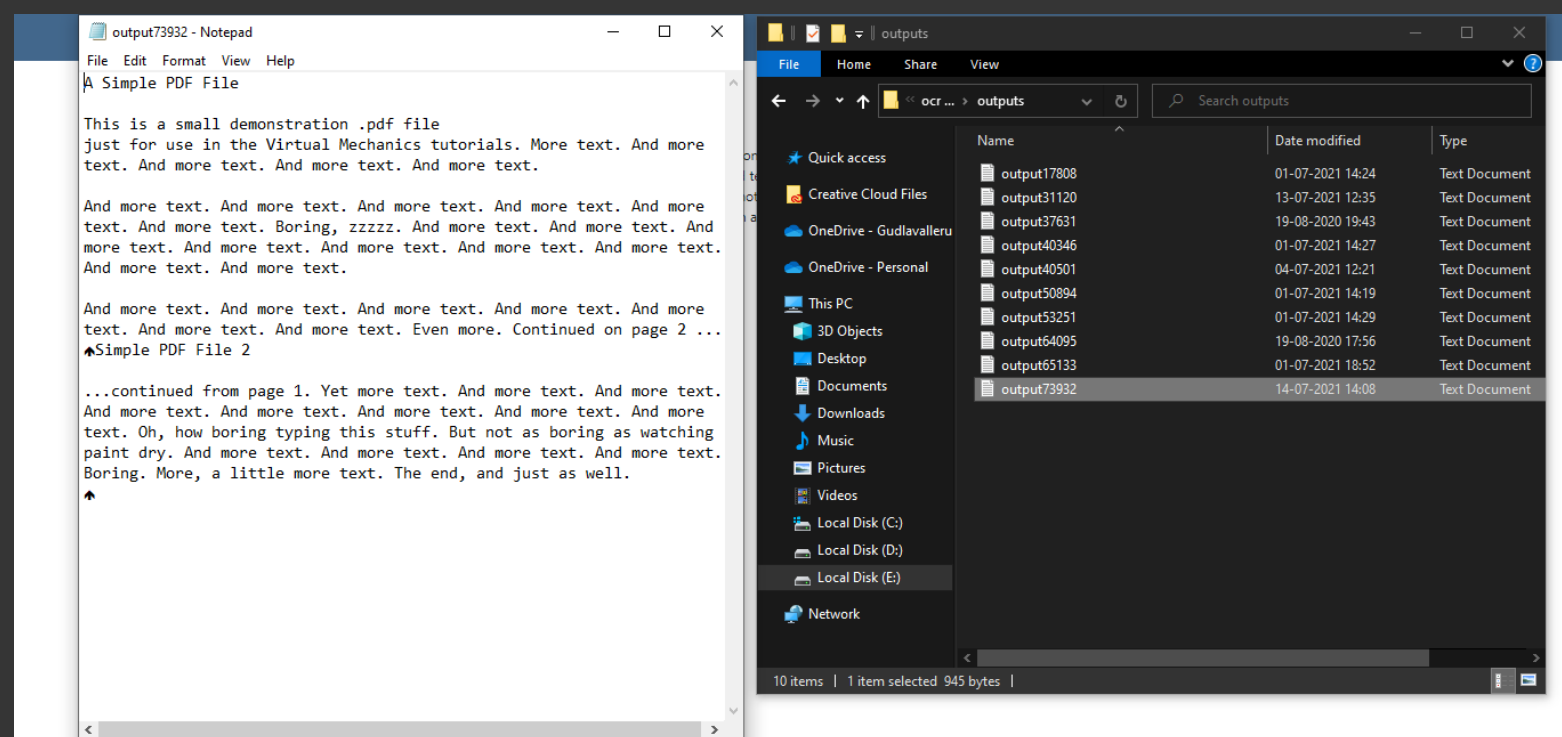
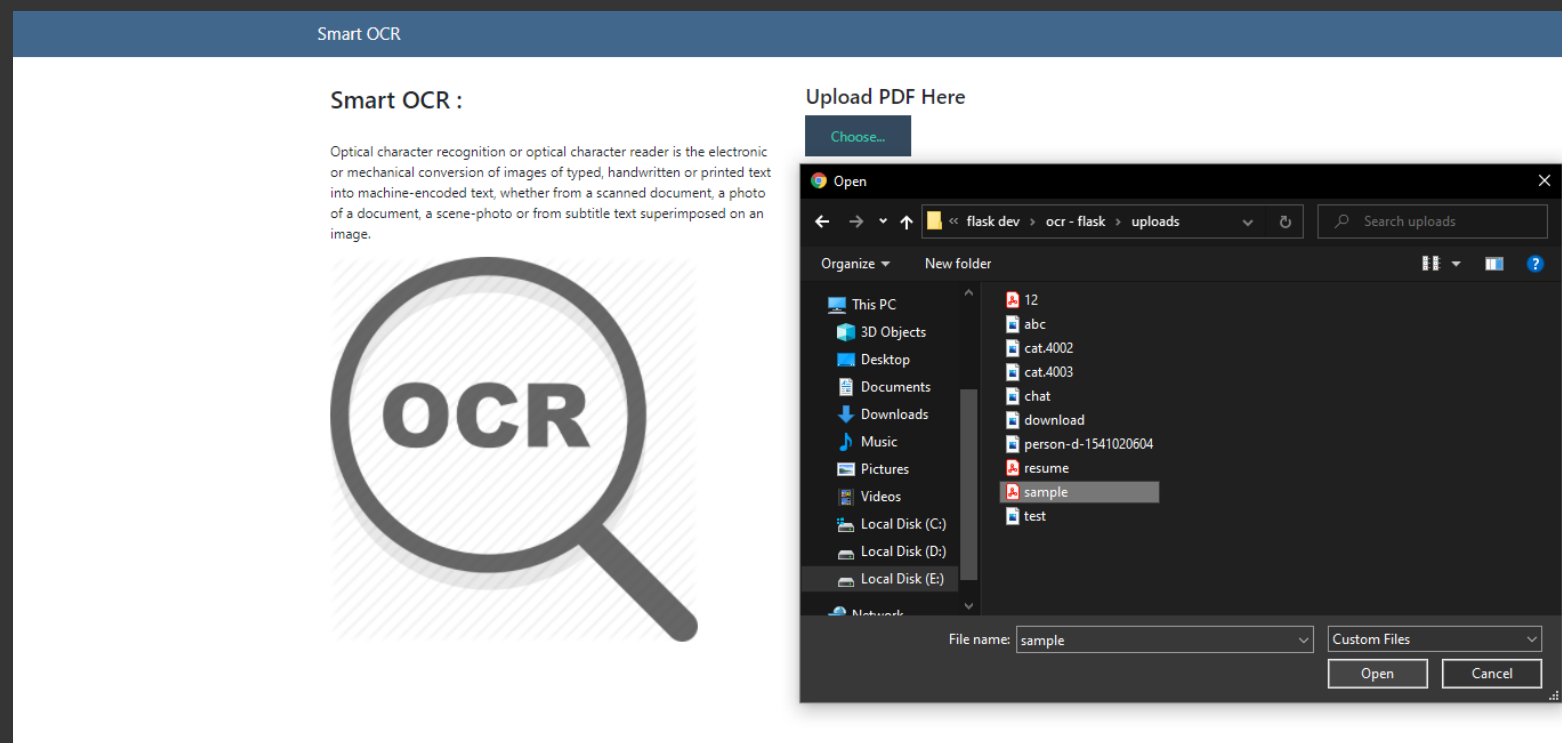
Smart OCR :

Optical character recognition or optical character reader is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image.



Upload PDF Here

Choose...



- Yes! Our Flask application has been able to integrate the OCR functionality and display the text on the browser. This makes it easier to process images instead of running commands on the CLI every time we have a new image to process.



# 7. Advantages and Disadvantages

## **Advantages of Optical character Reader (OCR) :**

Following are the advantages or advantages of OCR :

- Information of OCR can be readable with high degree of accuracy. Flatbed scanners are very accurate and may produce reasonably top quality images.
- Processing of OCR information is fast. Large quantities of text are often input quickly.
- A paper based form are often became an electronic form which is straightforward to store or send by mail.
- It is cheaper than paying someone amount to manually enter great deal of text data. Moreover it takes less time to convert within the electronic form.
- The latest software can re-create tables also as original layout.
- This process is much faster as compared to the manual typing the information into the system
- Advanced version can even Re create tables, columns and even produce sites.

## **Disadvantages of Optical character Reader (OCR) :**

Following are the drawbacks or disadvantages of OCR :

- OCR text works efficiently with the printed text only and not with handwritten text. Handwriting must be learnt by the pc.
- OCR systems are expensive.
- There is the need of lot of space required by the image produced.
- The quality of the image can be lose during this process.
- Quality of the ultimate image depends on quality of the first image.
- All the documents got to be checked over carefully then manually corrected.
- Not 100% accurate, there are likely to be some mistakes made during the method.
- Not worth doing for little amounts of text.

## 8.Applications

- Previously, digitization of documents was achieved by manually typing the text on the computer. Through OCR, this process is made easier as the document can be scanned, processed and the text extracted and stored in an editable form such as a word document.
- If you have a document scanner on your phone, such as Adobe Scan, you have probably encountered OCR technology in use.
- Airports can also use OCR to automate the process of passport recognition and extraction of information from them.
- Other uses of OCR include automation of data entry processes, detection, and recognition of car number plates.

## 9.Conclusion

- Through Tesseract and the Python-Tesseract library, we have been able to scan images and extract text from them. This is Optical Character Recognition and it can be of great use in many situations.
- We have built a scanner that takes an image and returns the text contained in the image and integrated it into a Flask application as the interface. This allows us to expose the functionality in a more familiar medium and in a way that can serve multiple people simultaneously.

## 10.Future scope

OCR is finally moving away from just seeing and matching. Driven by deep learning, it's entering a new phase where it first recognizes scanned text, then makes meaning of it. The competitive edge will be given to the software that provides the most powerful information extraction and highest-quality insights.

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.
- Many applications exist where it would be desirable to read handwritten entries.
- Reading handwriting is a very difficult task considering the diversities that exist in ordinary penmanship. However, progress is being made.

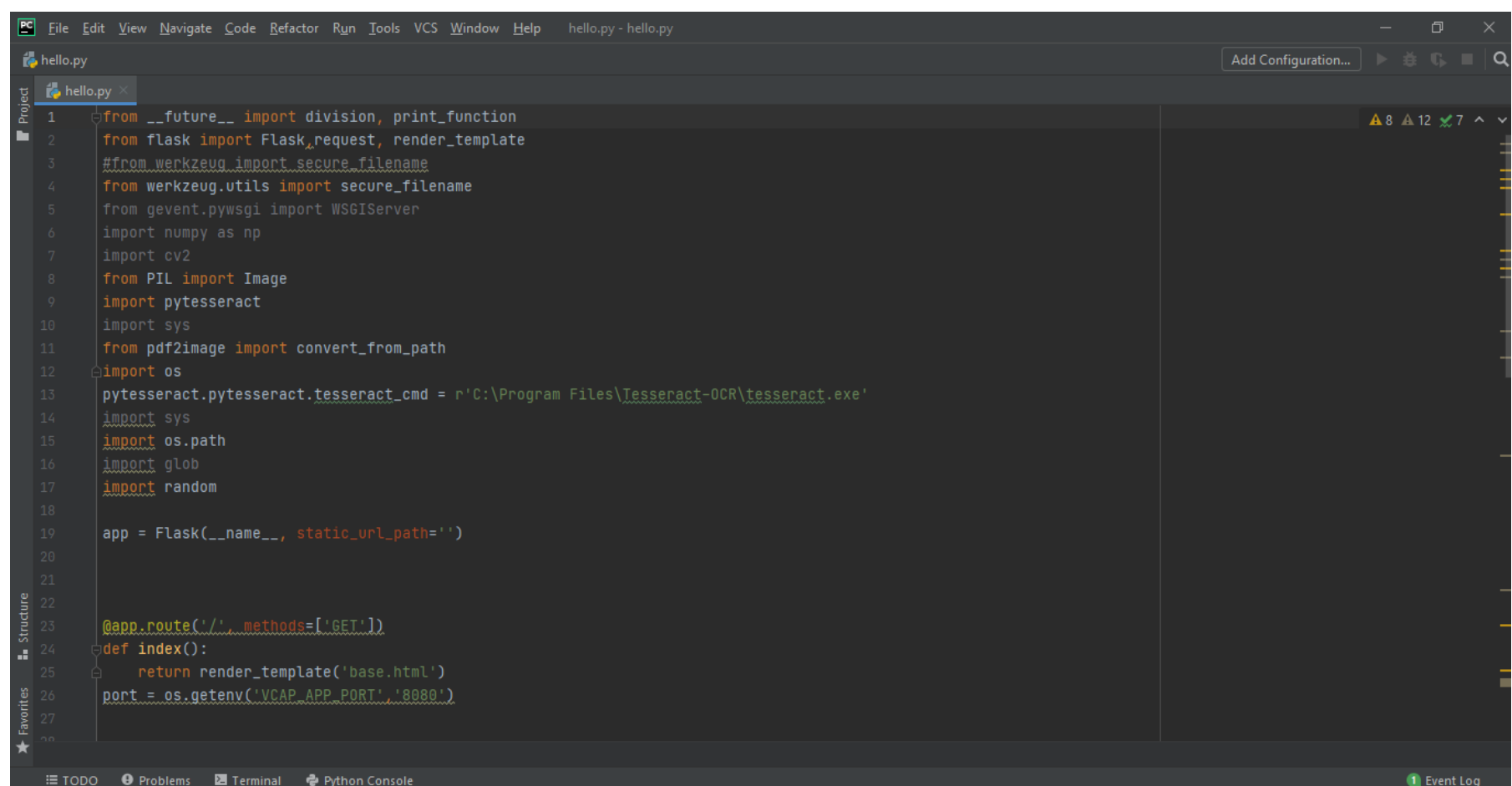
# 11. Bibliography

References of previous works or websites visited/referred for analysis about the project, solution previous findings etc.

- [https://www.youtube.com/watch?v=lj4l\\_CvBnt0](https://www.youtube.com/watch?v=lj4l_CvBnt0) - for flask development
- <https://nanonets.com/blog/ocr-with-tesseract/>

# 12. Appendix

source code:



```
1 from __future__ import division, print_function
2 from flask import Flask, request, render_template
3 #from werkzeug import secure_filename
4 from werkzeug.utils import secure_filename
5 from gevent.pywsgi import WSGIServer
6 import numpy as np
7 import cv2
8 from PIL import Image
9 import pytesseract
10 import sys
11 from pdf2image import convert_from_path
12 import os
13 pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
14 import sys
15 import os.path
16 import glob
17 import random
18
19 app = Flask(__name__, static_url_path='')
20
21
22
23 @app.route('/', methods=['GET'])
24 def index():
25     return render_template('base.html')
26 port = os.getenv('VCAP_APP_PORT', '8080')
27
28
```

```
hello.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help hello.py - hello.py
Add Configuration...
hello.py
28
29 @app.route('/predict', methods=['GET', 'POST'])
30 def upload():
31     if request.method == 'POST':
32         f = request.files['image']
33         basepath = os.path.dirname(__file__)
34         file_path = os.path.join(
35             basepath, 'uploads', secure_filename(f.filename))
36         f.save(file_path)
37         PDF_file = file_path
38         pages = convert_from_path(PDF_file, 500)
39         image_counter = 1
40         for page in pages:
41             filename = "page_"+str(image_counter)+".jpg"
42             page.save(filename, 'JPEG')
43             image_counter = image_counter + 1
44         filelimit = image_counter-1
45         # Creating a text file to write the output
46         basepath = os.path.dirname(__file__)
47         file_path2 = os.path.join(
48             basepath, 'outputs', "output"+str(random.randint(1, 100000))+".txt")
49         f = open(file_path2, "a")
50         for i in range(1, filelimit + 1):
51             filename = "page_"+str(i)+".jpg"
52             text = str((pytesseract.image_to_string(Image.open(filename))))
53             text = text.replace('-\n', '')
54             f.write(text)
55         f.close()
56         return file_path2
57
58
59 if __name__ == '__main__':
60     app.secret_key = os.urandom(12)
61     app.run(debug=True, host='0.0.0.0', port=port)
62
63
TODO Problems Terminal Python Console Event Log
```

# UI output:

