https://www.indianchemist.com/images/iclogotext.png

**OCR Documentation**

Contents

[Aim 2](#_Toc81410657)

[Languages and libraries Used 2](#_Toc81410658)

[Programming Language used 2](#_Toc81410659)

[Libraries used 3](#_Toc81410660)

[Process 3](#_Toc81410661)

[Code 3](#_Toc81410662)

[Deploy Flask on IIS 10](#_Toc81410663)

[Result 10](#_Toc81410664)

[Input 11](#_Toc81410665)

[Output 12](#_Toc81410666)

[Appendix 13](#_Toc81410667)

[Full Code 13](#_Toc81410668)

# Aim

The purpose of this project is to extract the named entities from the Doctor’s prescriptions that will be uploaded by users on the IndianChemist website and the mobile application.

The following variables/entities are being extracted from the prescriptions-

* Doctor Name
* Email
* Mobile No.
* Landline No.
* Hospital Name
* Address,
* Department
* Closing Day

# Languages and libraries Used

Programming Language used

* Python

## Libraries used

* **For Reading the user input image (prescription) from URL** – PIL (Python Imaging Library)
* **For Brightening the input image** - PIL (Python Imaging Library)
* **For image processing (resizing, grayscale conversion, thresholding)** – CV2 (Computer Vision library)
* **For raw text extraction** – Tesseract 5.0 (OCR engine)
* **For named entities extraction from raw text** – Python Regular Expressions (Regex), and other python built-in functions (split, replace, etc.)

# Process

Following processes are done in the below described chronological order-

1. Reading the image from the respective URL
2. Enhancing the brightness of the above read image
3. Resizing the image to get the most optimum output raw text
4. Converting the resized image to Grayscale from RGB (for thresholding)
5. Thresholding the image to convert it into a binary image (black and white)
6. Use the Tesseract OCR engine (5.0) to extract the raw data from the thresholded image
7. Use the Python Regex function and other python functions to transform the raw data and get the named entities from the data
8. Make a flask app and deploy on IIS

# Pre-requisites

1. Download and install python 3.8
2. Download and install Tesseract 5.0
3. Use the following commands in the command prompt to install the respective libraries in the python environment.

* **pip install numpy**
* **pip install opencv-python**
* **pip install Pillow**
* **pip install pytesseract**
* **pip install Flask**
* **pip install urllib3**
* **pip install requests**
* **pip install jsonlib**

# Code

from flask import make\_response

from flask import request

from flask import render\_template

from flask import Flask, jsonify

from flask import abort

import re

import urllib

import numpy as np

import cv2

import json

import PIL

from PIL import Image

import requests

import pytesseract

import numpy as np

from PIL import Image, ImageEnhance

pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files (x86)\Tesseract-OCR\tesseract.exe'

Importing the required libraries for reading the input, processing the input prescription, Tesseract (OCR engine used), and flask for making web tool.

**from flask import make\_response** -  *for adding additional HTTP headers to a response within a view's code.*

**from flask import request** -  *Flask request object contains the data that the client has sent to your app – i.e. the URL parameters*

**from flask import render\_template** *- render\_template is used to generate output from a template file based on the Jinja2 engine that is found in the application's templates folder.*

**from flask import Flask, jsonify** *- jsonify serializes data to JavaScript Object Notation (JSON) format*

**from flask import abort** *- aborts a request with an HTTP error code*

**import re** – *used for importing python regular expressions*

**import urllib** *- Urllib module is the URL handling module for python. It is used to fetch URLs*

**import numpy as np** – *importing the numpy library*

**import cv2** – *importing the open CV computer vision library*

**import json** - *used to work with JSON data*

**import PIL** - *importing the Python imaging library*

**from PIL import Image** -  *provides the python interpreter with image editing capabilities. The Image module provides a class with the same name which is used to represent a PIL image. The module also provides a number of factory functions, including functions to load images from files, and to create new images.*

**import requests** - *the requests module allows you to send HTTP requests using Python.*

**import pytesseract** – *It imports the python wrapper for tesseract OCR engine.*

**from PIL import Image, ImageEnhance** - *The ImageEnhance module contains a number of classes that can be used for image enhancement.*

pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files (x86)\Tesseract-OCR\tesseract.exe' - *is to declare the path for the Tesseract OCR engine in the system.*

url = request.json['url']

image=Image.open(requests.get(url, stream=True).raw)

enhancer = ImageEnhance.Brightness(image)

img\_light = enhancer.enhance(.90)

open\_cv\_image = np.array(img\_light)

scale\_percent = 200 # percent of original size

width = int(open\_cv\_image.shape[1] \* scale\_percent / 100)

height = int(open\_cv\_image.shape[0] \* scale\_percent / 100)

dim = (width, height)

large = cv2.resize(open\_cv\_image,dim, interpolation = cv2.INTER\_CUBIC)

#large = cv2.resize(open\_cv\_image, None, fx=4, fy=4, interpolation=cv2.INTER\_CUBIC)

gray1 = cv2.cvtColor(large , cv2.COLOR\_BGR2GRAY)

ret, thresh2 = cv2.threshold(gray1, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

image\_to\_text1 = pytesseract.image\_to\_string(large, lang='eng')

image\_to\_text2 = pytesseract.image\_to\_string(thresh2 , lang='eng')

**url = request.json['url']** – *Requesting the URL*

**image=Image.open(requests.get(url, stream=True).raw)** – *opening the image read from URL*

**enhancer = ImageEnhance.Brightness(image)** – *calling the function for brightness enhancement*

**img\_light = enhancer.enhance(.90)** – *increasing the brightness of the input image*

**open\_cv\_image = np.array(img\_light)** – *converting the image to numpy array*

**scale\_percent = 200** - *percent of original size*

**width = int(open\_cv\_image.shape[1] \* scale\_percent / 100)** – *converting width to integer*

**height = int(open\_cv\_image.shape[0] \* scale\_percent / 100)** – *converting height to integer*

**dim = (width, height)** – *declaring dim as a variable*

**large = cv2.resize(open\_cv\_image,dim, interpolation = cv2.INTER\_CUBIC)** - *Resizing the image*

**gray1 = cv2.cvtColor(large , cv2.COLOR\_BGR2GRAY)** - *Converting the resized image to Grayscale from RGB (for thresholding)*

**ret, thresh2 = cv2.threshold(gray1, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)** -  *thresholding the image to convert it into a binary image (black and white)*

**image\_to\_text1, image\_to\_text2** -  *we have used the enlarged image and the thresholded image to extract the raw text using the Pytesseract ‘Imagetostring’ function that will convert the image to unstructured text.*

dx=[]

if len(image\_to\_text1)>len(image\_to\_text2):

dx.append(image\_to\_text1)

else:

dx.append(image\_to\_text2)

rx=str(dx)

image\_to\_text=rx.replace('\n',' ').replace('\\n',' ')

*Now we compare the length of the strings in the output from the enlarged and threshold image.*

*The one in having more length becomes our raw text for the extraction of named entities.*

*This raw text is further cleaned by replacing the new line characters (\n).*

dr = re.findall(r'(Dr\.\s\S+\s\S+\b)', image\_to\_text , re.IGNORECASE)

email = re.findall(r'[\w\.-]+@[\w\.-]+', image\_to\_text )

phones = re.findall('(?:\+ \*)?\d[\d\- ]{6,}\d', image\_to\_text)

phone\_number=[phone.replace('-', '').replace(' ', '') for phone in phones]

MobileNo = [phone for phone in phone\_number if len(phone)==10 or len(phone)==13]

LandLineNo = [phone for phone in phone\_number if len(phone)==8 or len(phone)==11 ]

time1 = re.findall(r'\s(\d{1,2}\.\d{1,2}\s?(?:AM|PM|am|pm|A.M|P.M|a.m|p.m|A.M.|P.M.|pM.|Noon))', image\_to\_text)

time2 = re.findall(r'\s(\d{1,2}\:\d{1,2}\s?(?:AM|PM|am|pm|A.M|P.M|a.m|p.m|A.M.|P.M.|pM.|Noon))', image\_to\_text)

hospital = re.findall(r'(\w\S+\s+)(?=hospital|clinic|hos|hosp|hospi|hospit|clini|clinics|hospitals){1}' , image\_to\_text , re.IGNORECASE)

#splitby\n for hospital

Hospital\_name\_by\_split=[]

hosp = ['Hospital' , 'Clinic', 'hospital','clinic','clinics','centre' ,'Hospit@l' , 'Hospitl' , 'Nursing' , 'Home']

for x in image\_to\_text1.split('\n'):

a\_match = [True for match in hosp if match in x]

if True in a\_match:

Hospital\_name\_by\_split.append(x)

Hospital\_name\_by\_split.extend(hospital)

else:

Hospital\_name\_by\_split.append(hospital)

break

data=[]

replacedtext = image\_to\_text1.replace('\n','.')

#for x in replacedtext.split('.'):

# if [str.isdigit() for str in x].count(True)==(6|7):

# data=x

# Address = data

AddressbyString= re.findall(r'(\w\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+)(?=Andhra Pradesh|Arunachal Pradesh|Assam|Bihar|Chhattisgarh|Goa|Gujarat|Haryana|Himachal Pradesh|Jharkhand|Karnataka|Kerala|Madhya Pradesh|Maharashtra|Manipur|Meghalaya|Mizoram|Nagaland|Odisha|Punjab|Rajasthan|Sikkim|Tamil Nadu|Telangana|Tripura|Uttar Pradesh|Uttarakhand|West Bengal|Andaman and Nicobar Islands|Chandigarh|Dadra & Nagar Haveli and Daman & Diu|Delhi|Jammu and Kashmir|Lakshadweep|Puducherry|Ladakh|Hyderabad|Itanagar|Dispur|Patna|Raipur|Panaji|Gandhinagar|Chandigarh|Shimla|Ranchi|Bengaluru|Bangalore|Thiruvananthapuram|Bhopal|Mumbai|Imphal|Shillong|Aizawl|Kohima|Bhubaneswar|Jaipur|Gangtok|Chennai|Hyderabad|Agartala|Lucknow|Dehradun|Kolkata|Port Blair|Daman|New Delhi|Srinagar|Kavaratti|Pondicherry|Leh)', image\_to\_text)

Department = re.findall(r'(\w\S+\s+)(?=Department){3}', image\_to\_text)

#splitby\n for Department

Department\_by\_split=[]

#Departments = ['Department']

for x in image\_to\_text1.split('\n'):

#a\_match = [True for match in Departments if match in x]

#if True in a\_match:

if 'Department' in x:

Department\_by\_split.append(x)

Department\_by\_split.extend(Department)

else:

Department\_by\_split=Department

break

ClosingDay = re.findall(r'(\w\S+\s+)(?=Closed){1}' , image\_to\_text)

*In the Code above, we have used the python regular expressions with different wildcards and other python built in functions to extract the following named entities from the raw text.*

1. *Doctor Name – Identified by the string containing ‘Dr.’*
2. *Email id- Identified by the string containing ‘@’*
3. *Mobile No.- Identified by the numeric strings of length 10*
4. *Mobilewith91 - Identified by the numeric strings of length 13*
5. *Land Line No. - Identified by the numeric strings of length 8*
6. *Land line No. with code - Identified by the numeric strings of length 11*
7. *Hospital/Clinic/Nursing Home Name – Identified by extracting strings prior to the words like ‘Hospital’ , ‘Clinic’ , ‘Nursing Home Name’*
8. *Hospital Name by Split – Identified by the string containing words like ‘Hospital’ , ‘Clinic’ , ‘Nursing Home Name’ out of all the strings split by ‘\n’*
9. *time1 (separated by ‘.’) – extracted by numeric strings having ‘.’ between them*
10. *time2 (separated by ‘:’) - extracted by numeric strings having ‘:’ between them*
11. *Address- Identified by extracting strings prior to the Indian Cities.*
12. *Address by String- Identified by the string containing 6 digit numeric pin-code out of all the strings split by ‘\n’*
13. *Department - Identified by extracting strings after the word ‘Department’*
14. *Department by split- Identified by the string containing word ‘Department’ out of all the strings split by ‘\n’*
15. *Closing Day - Identified by extracting strings prior to word ‘Closed’*

myjson={

"Status":200,

"data":image\_to\_text,

"DoctorName":dr,

"Email": email,

"MobileNo":MobileNo,

"LandLineNo":LandLineNo,

"HospitalName":Hospital\_name\_by\_split,

#"HospitalName2":Hospital\_name\_by\_split,

#"time1":time1,

#"time2":time2,

#"Address":Address,

"Address":AddressbyString,

"Department":Department\_by\_split,

"ClosingDay":ClosingDay

}

*Now we have created a json and written down the {key: value} pairs for our extracted named entities.*

# Deploy Flask on IIS

**Step 1: Installing wfastcgi package in the python environment.**

Download the wfastcgi package from the link below

[An IIS-Python bridge based on WSGI and FastCGI.pypi.org](https://pypi.org/project/wfastcgi/" \t "_blank)

[The contents of the folder should look like this.](https://pypi.org/project/wfastcgi/" \t "_blank)

[](https://pypi.org/project/wfastcgi/" \t "_blank)

**Step 2: Enabling wfastcgi and configuring Fast CGI Settings in IIS**

The first step in enabling wfastcgi is to fire the wfastcgi-enable.exe placed in the ***python environment / scripts***folder

https://miro.medium.com/max/331/1*vyQpwUnle5TiMsTNsv0nZg.pngUse the following command in command prompt as administrator mode-

# https://miro.medium.com/max/366/1*mOAmJZ3WSwNk4_VeClbuIg.png

https://miro.medium.com/max/700/1*aiQp1nxI9d_ExR5d5CPZ4Q.pngYou should get the following message .

**Enabling FastCGI settings for IIS**

Click on Fast CGI Settings. Note that the FastCGI settings are done at a server level . So look at your IIS server’s root.

# https://miro.medium.com/max/700/1*dV9FEpH45aFHrSPZ406xSQ.png

This should show the following entry.

# https://miro.medium.com/max/700/1*1NoQt_CgHMUAu9RznKbZyw.png

Take a look at the arguments. You might get something like your python environment site packages/wfastcgi-3.0.0-py3.6.egg-info/wfastcgi.py.

With this we have successfully configured IIS to host python scripts.

**Step 3: Setting up the flask app on IIS.**

a. First add a new application in IIS with the name “FlaskRedirect”. You can name it anything.

# https://miro.medium.com/max/158/1*c69KDztyXO1puLbkhEhoTg.png

Now we need to get the contents in virtual directory for this application.

**b. myapp.py**

This is the python script file

**c.web.config**

Take a look at the parameters. Ensure that your web config file is updated to have the relevant entries in sync with your path.

scriptProcessor = Is the same as the entries that I had defined in the Fast CGI Settings

WSGI\_HANDLER = myapp.app ( where myapp comes from my flask script file name myapp.py )

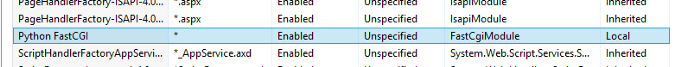
PYTHONPATH = is the virtual directory where I’m keeping my flask script and config file.

d. Setting up the handler mappings for the application.

Now it’s time to fire up the application and test. Before that we have one last thing to check.

And that’s the handler mapping. Although the web.config file has the handler mapping it is a good idea to ensure that it is there in your site settings.

# https://miro.medium.com/max/700/1*MBaU2aQT7jC1_JyoFjKzGg.png

Look for a mapping “Python FastCGI” .

Let’s check the settings now by double clicking on the mapping. It should look something like this.

# https://miro.medium.com/max/461/1*eH3Pn7Azu3_TATUnCv-G8Q.png

As a last step click on the Request Restrictions button and ensure the check box is **“un-checked”**

# https://miro.medium.com/max/461/1*sP8KQrv0yUH4G3srqMmySQ.png

Now, open URL on Chrome

# https://miro.medium.com/max/449/1*aUk3D-4V9aQ_3-2KZsO83A.png

With that we have successfully hosted a flask app on IIS with Fast CGI .

For any queries related to Flask deployment, refer to the link below-

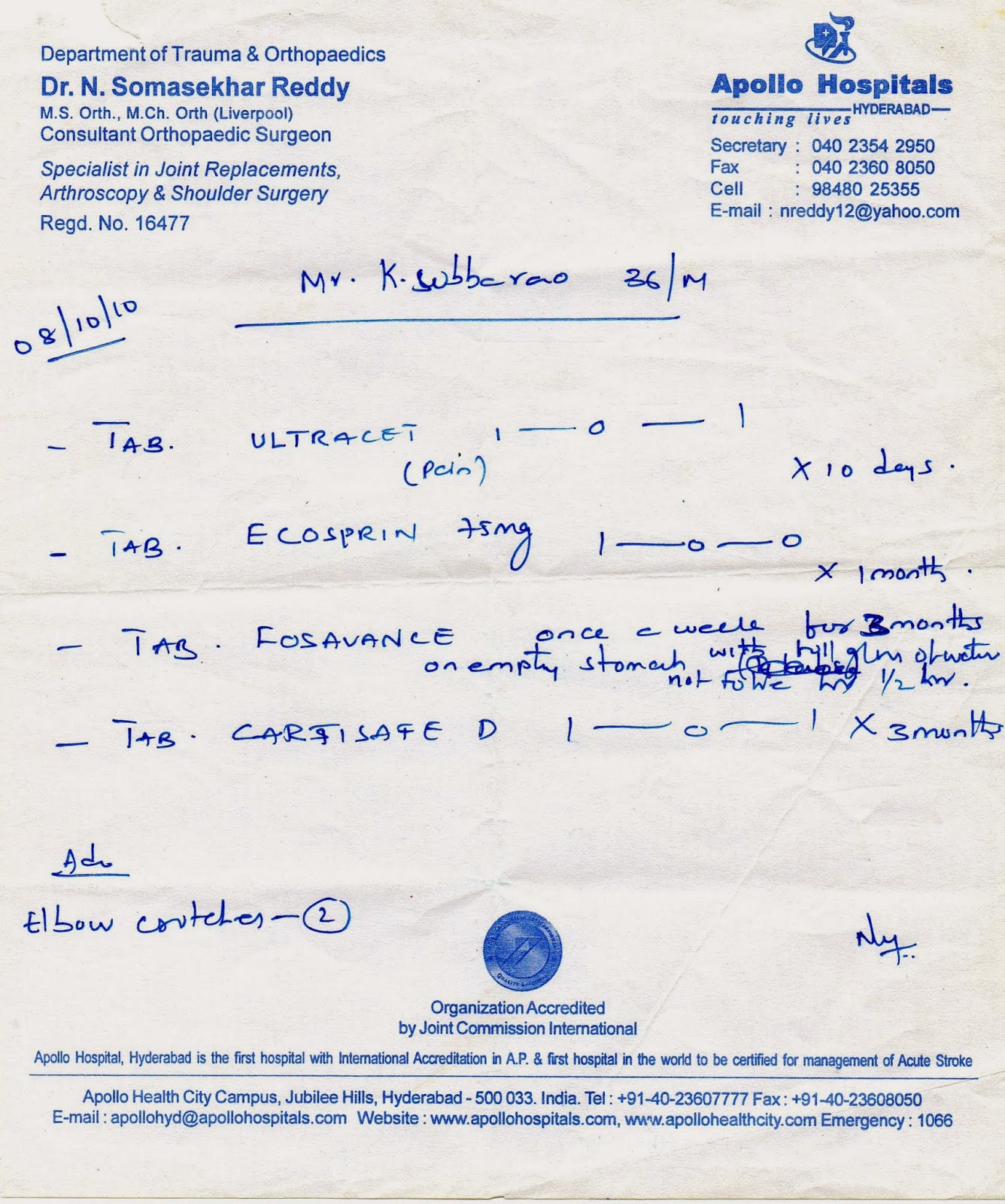
<https://rajesh-r6r.medium.com/deploying-a-python-flask-rest-api-on-iis-d8d9ebf886e9>

# Result

We have created a web application for our OCR that extracts the required named entities from the user’s input image. We have attached a input and output for one of the test case below.

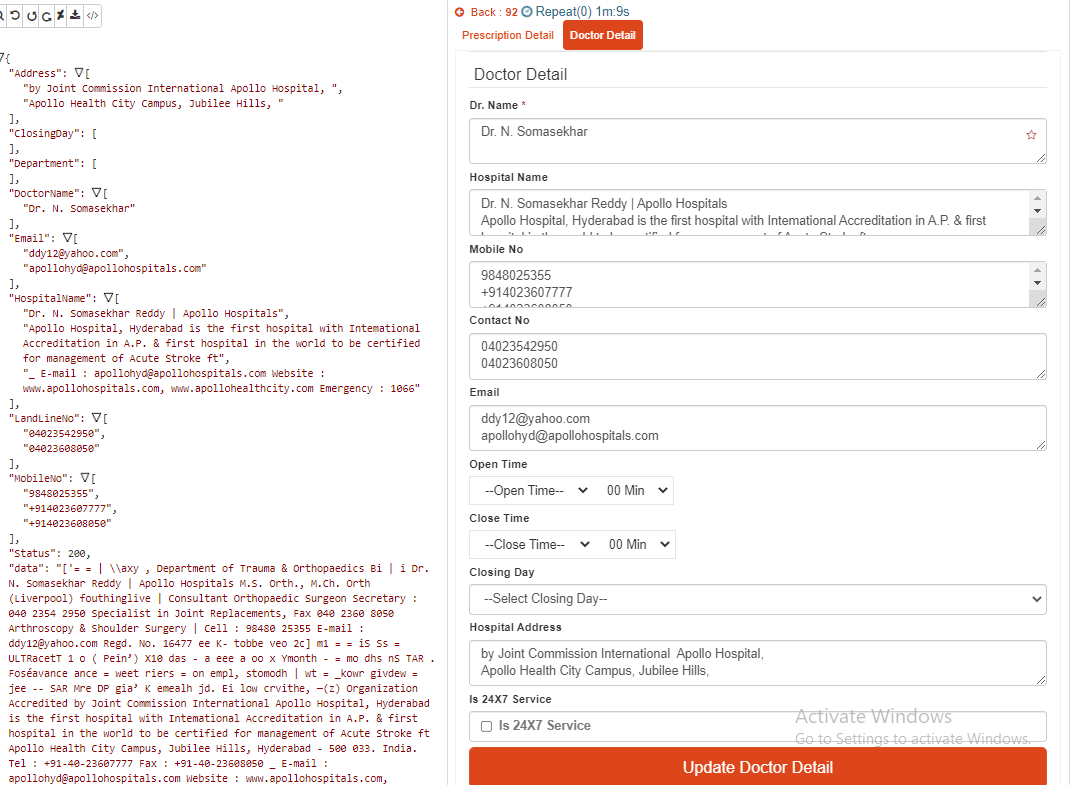
## 

## Input



## 

## Output



# Appendix

## Full Code

from flask import make\_response

from flask import request

from flask import render\_template

from flask import Flask, jsonify

from flask import abort

import re

import urllib

import numpy as np

import cv2

import json

import PIL

from PIL import Image

import requests

import pytesseract

import numpy as np

from PIL import Image, ImageEnhance

pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files (x86)\Tesseract-OCR\tesseract.exe'

start\_time = time.time()

app = Flask(\_\_name\_\_)

@app.route('/file\_upload', methods=['POST'])

def fileUpload():

try:

if not request.json or not 'url' in request.json:

abort(400)

url = request.json['url']

image=Image.open(requests.get(url, stream=True).raw)

enhancer = ImageEnhance.Brightness(image)

img\_light = enhancer.enhance(.90)

open\_cv\_image = np.array(img\_light)

scale\_percent = 200 # percent of original size

width = int(open\_cv\_image.shape[1] \* scale\_percent / 100)

height = int(open\_cv\_image.shape[0] \* scale\_percent / 100)

dim = (width, height)

large = cv2.resize(open\_cv\_image,dim, interpolation = cv2.INTER\_CUBIC)

#large = cv2.resize(open\_cv\_image, None, fx=4, fy=4, interpolation=cv2.INTER\_CUBIC)

gray1 = cv2.cvtColor(large , cv2.COLOR\_BGR2GRAY)

ret, thresh2 = cv2.threshold(gray1, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

image\_to\_text1 = pytesseract.image\_to\_string(large, lang='eng')

image\_to\_text2 = pytesseract.image\_to\_string(thresh2 , lang='eng')

dx=[]

if len(image\_to\_text1)>len(image\_to\_text2):

dx.append(image\_to\_text1)

else:

dx.append(image\_to\_text2)

rx=str(dx)

image\_to\_text=rx.replace('\n',' ').replace('\\n',' ')

dr = re.findall(r'(Dr\.\s\S+\s\S+\b)', image\_to\_text , re.IGNORECASE)

email = re.findall(r'[\w\.-]+@[\w\.-]+', image\_to\_text )

phones = re.findall('(?:\+ \*)?\d[\d\- ]{6,}\d', image\_to\_text)

phone\_number=[phone.replace('-', '').replace(' ', '') for phone in phones]

MobileNo = [phone for phone in phone\_number if len(phone)==10 or len(phone)==13]

LandLineNo = [phone for phone in phone\_number if len(phone)==8 or len(phone)==11 ]

time1 = re.findall(r'\s(\d{1,2}\.\d{1,2}\s?(?:AM|PM|am|pm|A.M|P.M|a.m|p.m|A.M.|P.M.|pM.|Noon))', image\_to\_text)

time2 = re.findall(r'\s(\d{1,2}\:\d{1,2}\s?(?:AM|PM|am|pm|A.M|P.M|a.m|p.m|A.M.|P.M.|pM.|Noon))', image\_to\_text)

hospital = re.findall(r'(\w\S+\s+)(?=hospital|clinic|hos|hosp|hospi|hospit|clini|clinics|hospitals){1}' , image\_to\_text , re.IGNORECASE)

#splitby\n for hospital

Hospital\_name\_by\_split=[]

hosp = ['Hospital' , 'Clinic', 'hospital','clinic','clinics','centre' ,'Hospit@l' , 'Hospitl' , 'Nursing' , 'Home']

for x in image\_to\_text1.split('\n'):

a\_match = [True for match in hosp if match in x]

if True in a\_match:

Hospital\_name\_by\_split.append(x)

Hospital\_name\_by\_split.extend(hospital)

else:

Hospital\_name\_by\_split.append(hospital)

break

data=[]

replacedtext = image\_to\_text1.replace('\n','.')

#for x in replacedtext.split('.'):

# if [str.isdigit() for str in x].count(True)==(6|7):

# data=x

# Address = data

AddressbyString= re.findall(r'(\w\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+)(?=Andhra Pradesh|Arunachal Pradesh|Assam|Bihar|Chhattisgarh|Goa|Gujarat|Haryana|Himachal Pradesh|Jharkhand|Karnataka|Kerala|Madhya Pradesh|Maharashtra|Manipur|Meghalaya|Mizoram|Nagaland|Odisha|Punjab|Rajasthan|Sikkim|Tamil Nadu|Telangana|Tripura|Uttar Pradesh|Uttarakhand|West Bengal|Andaman and Nicobar Islands|Chandigarh|Dadra & Nagar Haveli and Daman & Diu|Delhi|Jammu and Kashmir|Lakshadweep|Puducherry|Ladakh|Hyderabad|Itanagar|Dispur|Patna|Raipur|Panaji|Gandhinagar|Chandigarh|Shimla|Ranchi|Bengaluru|Bangalore|Thiruvananthapuram|Bhopal|Mumbai|Imphal|Shillong|Aizawl|Kohima|Bhubaneswar|Jaipur|Gangtok|Chennai|Hyderabad|Agartala|Lucknow|Dehradun|Kolkata|Port Blair|Daman|New Delhi|Srinagar|Kavaratti|Pondicherry|Leh)', image\_to\_text)

Department = re.findall(r'(\w\S+\s+)(?=Department){3}', image\_to\_text)

#splitby\n for Department

Department\_by\_split=[]

#Departments = ['Department']

for x in image\_to\_text1.split('\n'):

#a\_match = [True for match in Departments if match in x]

#if True in a\_match:

if 'Department' in x:

Department\_by\_split.append(x)

Department\_by\_split.extend(Department)

else:

Department\_by\_split=Department

break

ClosingDay = re.findall(r'(\w\S+\s+)(?=Closed){1}' , image\_to\_text)

myjson={

"Status":200,

"data":image\_to\_text,

"DoctorName":dr,

"Email": email,

"MobileNo":MobileNo,

"LandLineNo":LandLineNo,

"HospitalName":Hospital\_name\_by\_split,

#"HospitalName2":Hospital\_name\_by\_split,

#"time1":time1,

#"time2":time2,

#"Address":Address,

"Address":AddressbyString,

"Department":Department\_by\_split,

"ClosingDay":ClosingDay

}

return jsonify(myjson)

except Exception as e:

myjson={

"Status": 400,

"Message": "type error:"+str(e)

}

return jsonify(myjson);

@app.errorhandler(400)

def not\_found(error):

return make\_response(jsonify({'error': 'Id is Not Found'}), 400)

@app.route('/ic/ocr', methods=['POST'])

def ocr\_task():

try:

if not request.json or not 'title' in request.json:

abort(400)

imageurl = request.json['imageurl']

myjson={

"Status": 200,

"Message": imageurl ,

"Data":"OCR"

}

return jsonify({'MasterMapperID': myjson})

except:

myjson={

"Status": 400,

"Message": "Error occured" ,

"Data":""

}

return jsonify(myjson);

@app.errorhandler(400)

def not\_found(error):

return make\_response(jsonify({'error': 'url not found'}), 400)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)