**Project I**

# Text Scanner (OCR)

**(Client server application)**

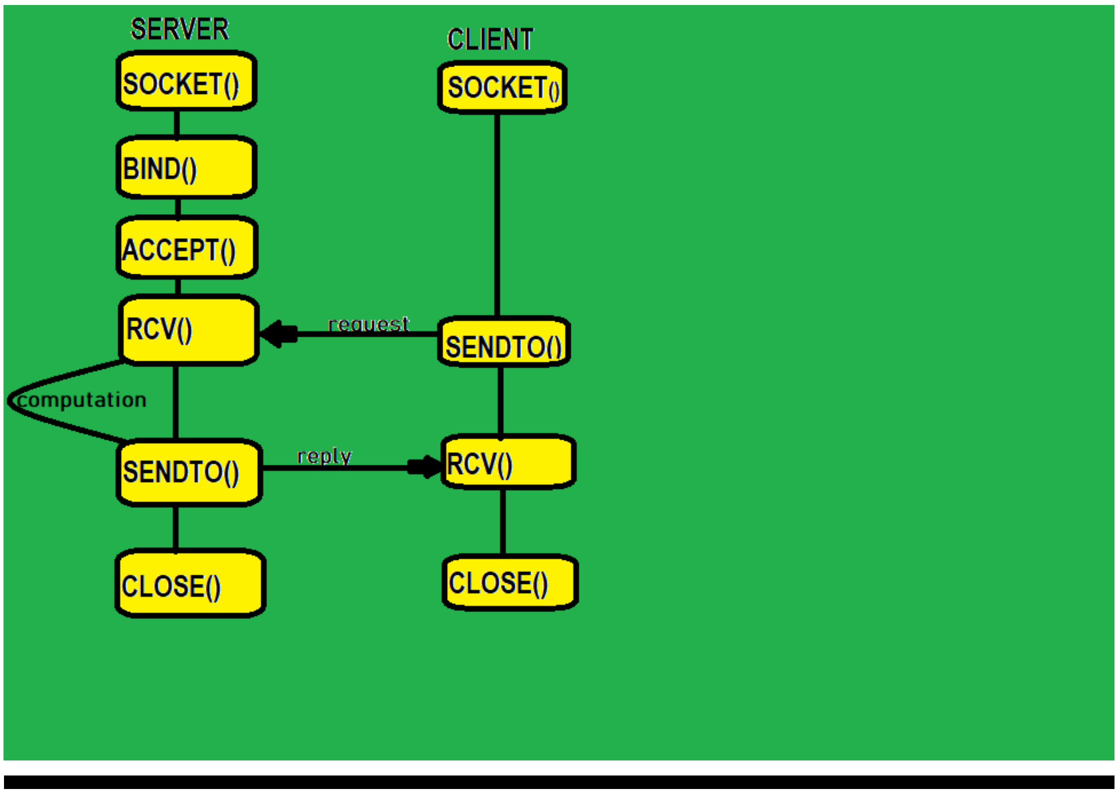
I20MA004: SOUMAYDEEP MANDAL

I20MA006: ABHIJEET BANSOD

I20MA029: URMIK BHAVSAR

I20MA030: RAHUL MAURYA

# Flowchart of Client server application



**This is flowchart of application in which server creates socket to receive request and reply to the client. Server Binds the IP address with port and accept the client connection on that address, after establishing the connection server receives and sends data to client as per client need. After that socket is closed.**

**More can be understood by pseudocode. (Written in .md file)**

# PSEUDOCODE

```
# Pseudocode


# for model.py
Make a function which returns the recognized texts from image and takes an
argument as image file:
    Define the lang to be recognized .
    Read the file to obtain the tuple containing the coordinates, text and
accuracy using easyocr.
    Set the coordinates.
    Read the file using cv2 as image.
    To mark the text iterate through the coordinates.
    Make rectangle on the recognized text passing arguments image ,x and y
coordinates ,color of border and width of border
    Now iterate the tuple and add only the text in empty string and return
that string.
    Use matplotlib to show the recognized text from image by marking them.
    *import all library that are needed*

# for server.py
Create socket for TCP Connection (SOCK_STREAM)
Get ip
Bind socket to a specific port and IP where clients can contact server
Listen for client connection
Accept the client connection
open the file in binary write mode (for recognizing the text)
Loop and receive the image_data in binary form from client
Write the image data in opened file
Close the file
Now got the image so use model.py to recognize the text from the image
Send the return (Which is text and accuracy) value of model.py to client
*import all needed libraries*
Close Socket

# for client.py
Create Socket for TCP Connection (SOCK_STREAM)
Get Ip
Connect to Server's address
Open file in binary read mode
```

```
Read file
Loop and send all the binary data to server
close the file
Now Receive recognized text from server using loop.
Decode the message
Print the message in form of Table with Text and Accuracy as Heading
*import all needed libraries*
Close Socket
```

# <u>CODE</u>

## model.py

```python
import easyocr,sys
import matplotlib.pyplot as plt
import cv2
import easyocr

def texts(file):

    # actual model working
    reader = easyocr.Reader(['en'])
    # results =tuple containing x and y coordinates ,text and accuracy of the
recognition
    results = reader.readtext(file)

    # now for marking the text and showing the image with marks
    cord = results[-1][0]
    image = cv2.imread(file)
    for (cord,text,accuracy) in results:
        # cooridinates of the text
        (x1, x2, y1, y2) = cord
        x1 = (int(x1[0]), int(x1[1]))
        x2 = (int(x2[0]), int(x2[1]))
        y1 = (int(y1[0]), int(y1[1]))
        y2 = (int(y2[0]), int(y2[1]))
    # for putting mark on text
        cv2.rectangle(image, x1, y1, (25, 25, 255), 9)
        plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

    # showing image with recognized text
    plt.savefig(file)

    # returning text and accuracy
    content = ''
    prob=''
    for item in results:
        content += item[1]+' '
        num=item[2]*100
        prob +=str('%.2f' % num)+' '

    return content,prob
```

## server.py

```
import socket,sys,model,io
from PIL import Image

# creating socket
server=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
ip=socket.gethostbyname(socket.gethostname())
print("Server is running")
server.bind((ip,8899))
server.listen(4)

# accepting the client request
client,address=server.accept()
print("Server is ready to accept data ...")
print("connected with client having",address)

# for binary data and io for input output operation
file_stream=io.BytesIO()

image_data=client.recv(2048)
# writing the image data
```

```python
while image_data:
    file_stream.write(image_data)
    image_data=client.recv(2048)
    if image_data==b'%DONE%' :
        break
# image converted
image=Image.open(file_stream)
filename=str(sys.argv[1])
# saving image
image.save(filename,format='JPEG')

msg,accuracy=model.texts(filename)
# print(accuracy)
check=True
while check==True:
    # sending reply
    client.send(msg.encode('utf-8'))
    client.send(accuracy.encode('utf-8'))
    check=False
# closing the socket
client.close()
```

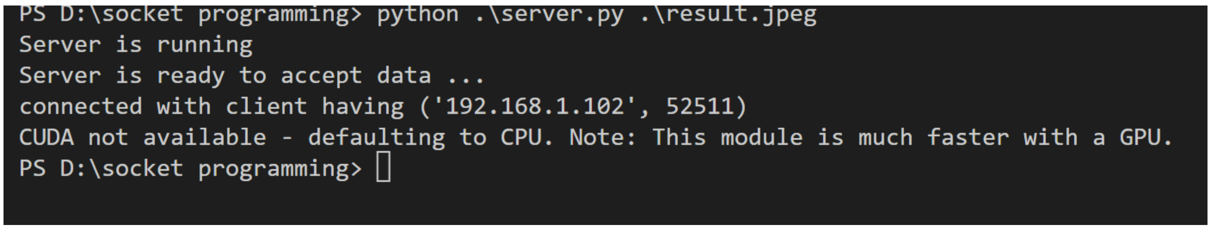
## client.py

```python
import socket,sys
from prettytable import PrettyTable
# creating socket
client=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
ip=socket.gethostbyname(socket.gethostname())
# connecting the client
client.connect((ip,8899))
print("client conected")


# read binary data from image file
file=open(sys.argv[1],'rb')
image_data=file.read(2048)
```

```
# send image data to server
while image_data :
    client.send(image_data)
    image_data=file.read(2048)
# closing the file
file.close()
# condition for loop exit
client.send(b"%DONE%")
check=True
# receiving the data
while(check==True):
    msg=client.recv(1024)
    accuracy=client.recv(1024)
    check=False
# making the text in list form
real_text=msg.decode('utf-8')
msg=(msg.decode('utf-8')).strip()
msg=msg.split(" ")
accuracy=(accuracy.decode('utf-8')).strip()
accuracy=accuracy.split(" ")


# for only text
print("****************************************************************")
print("Result of text Recognition is:\n")
print("****************************************************************")
print(real_text)
print("****************************************************************")
# for table form
print("Result in Table form")

# making table
myTable = PrettyTable(["Text", "Accuracy"])
for i in range(len(msg)):
    try:
        myTable.add_row([msg[i], accuracy[i]])
    except:
        IndexError
print(myTable)

# closing the sockt
client.close()
```

# OUTPUT

## Server



```
PS D:\socket programming> python .\server.py .\result.jpeg
Server is running
Server is ready to accept data ...
connected with client having ('192.168.1.102', 52511)
CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
PS D:\socket programming>
```

**PS D:\socket programming> python .\server.py .\result.jpeg**

**Server is running**

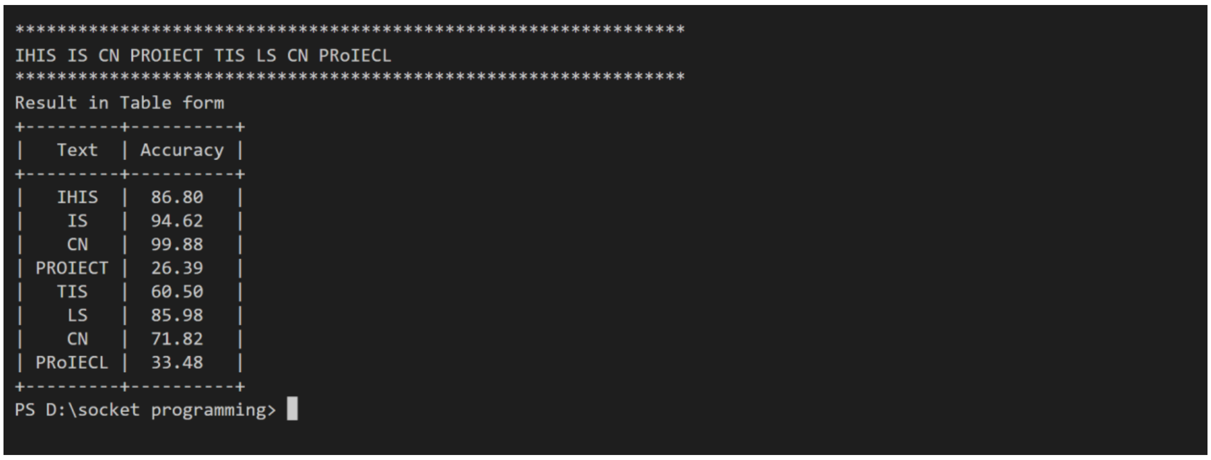
**Server is ready to accept data ...**

**connected with client having ('192.168.1.102', 52511)**

**CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.**

**PS D:\socket programming>**

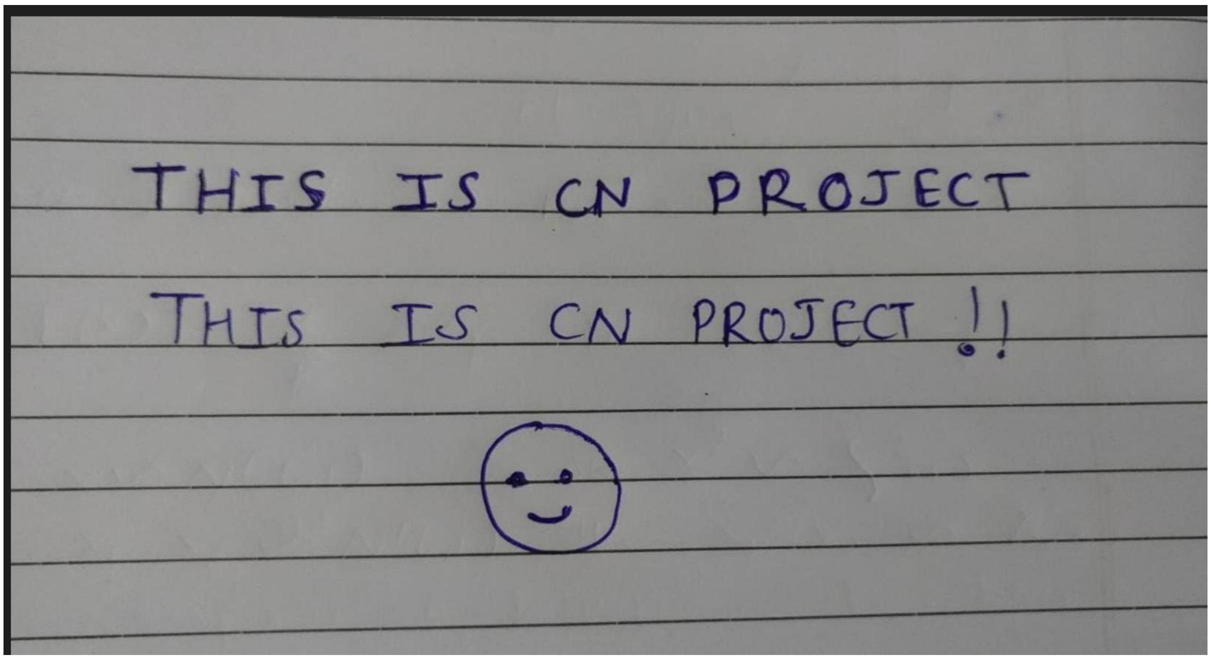
## Client



```
****************************************************************
IHIS IS CN PROIECT TIS LS CN PRoIECL
****************************************************************
Result in Table form
+---------+----------+
|   Text  | Accuracy |
+---------+----------+
|   IHIS  |  86.80   |
|    IS   |  94.62   |
|    CN   |  99.88   |
| PROIECT |  26.39   |
|   TIS   |  60.50   |
|    LS   |  85.98   |
|    CN   |  71.82   |
| PRoIECL |  33.48   |
+---------+----------+
PS D:\socket programming>
```

```
Result in text form:
********************************************************************
IHIS IS CN PROIECT TIS LS CN PRoIECL
********************************************************************
Result in Table form
Text       Accuracy
|  IHIS  |  86.80   |
|   IS   |  94.62   |
|   CN   |  99.88   |
| PROIECT |  26.39   |
|  TIS   |  60.50   |
|   LS   |  85.98   |
|   CN   |  71.82   |
| PRoIECL |  33.48   |
+---------+----------+
PS D:\socket programming>
```

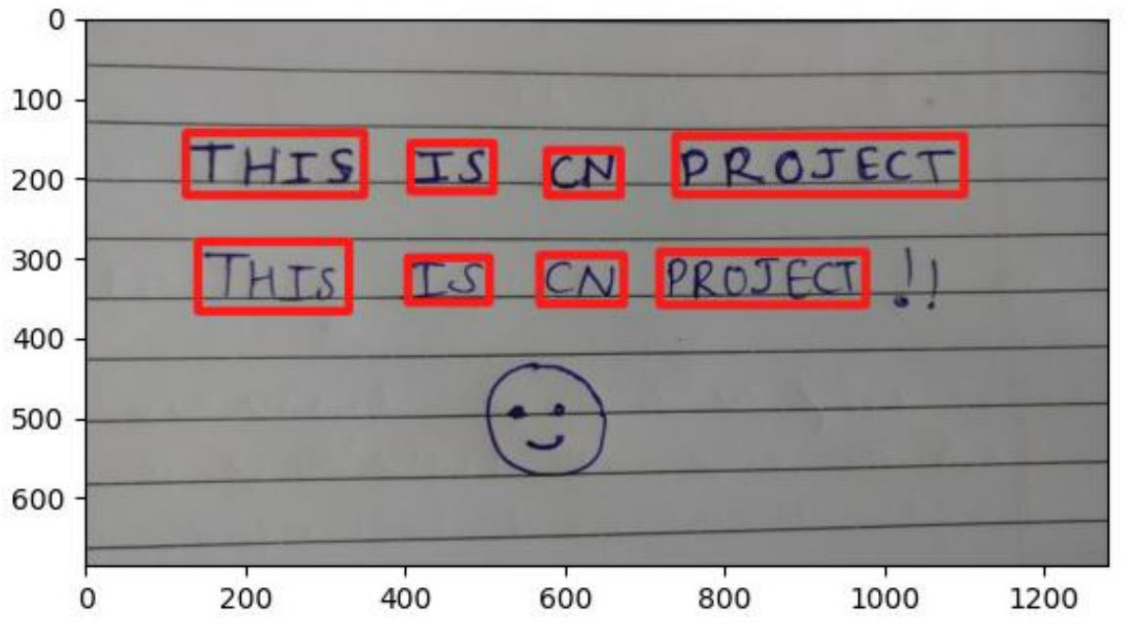
## Input Image



THIS IS CN PROJECT

THIS IS CN PROJECT !!

## Output Image

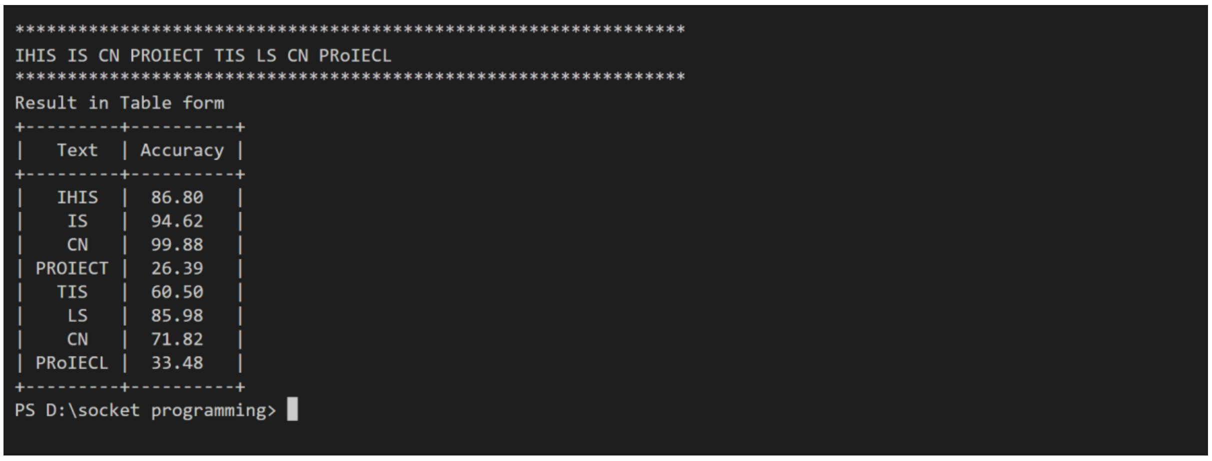


### EXAMPLE

- ABOVE IMAGE IS REAL EXAMPLE OF THIS APPLICATION WHERE IT DETECTS TEXTS WITH DIFFERENT ACCURACY

## ANALYZING THE OUTPUT

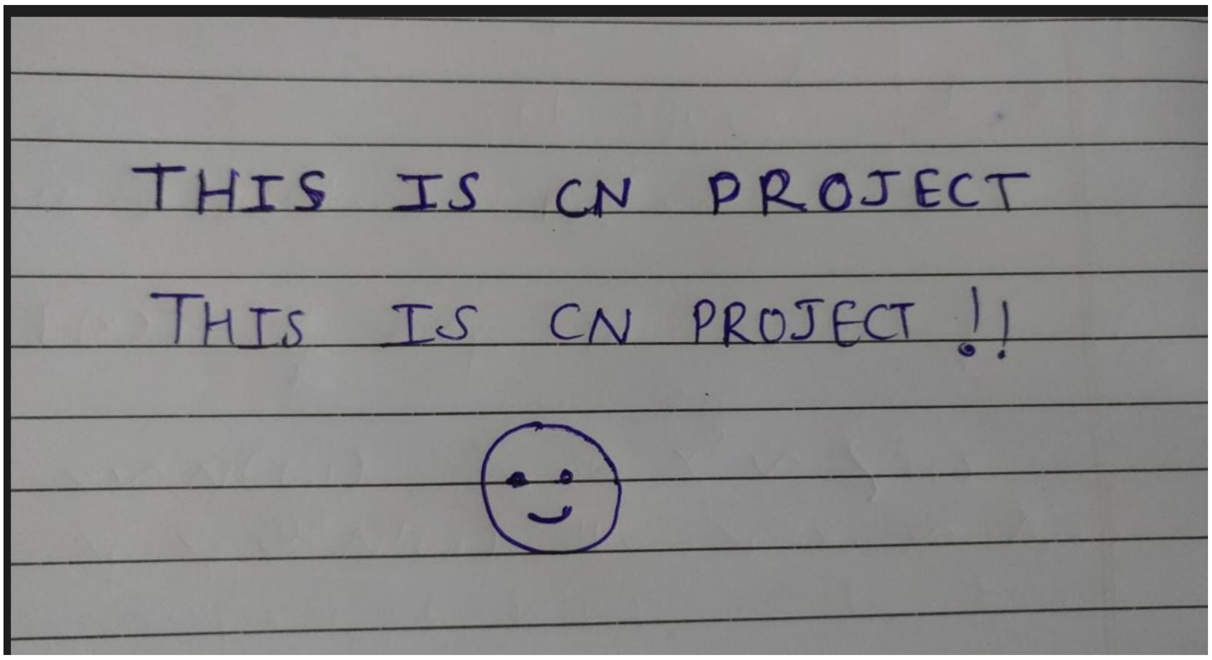
- **Client got output as given here**



```
*****************************************************************
IHIS IS CN PROIECT TIS LS CN PRoIECL
*****************************************************************
Result in Table form
+---------+----------+
|   Text  | Accuracy |
+---------+----------+
|   IHIS  |  86.80   |
|    IS   |  94.62   |
|    CN   |  99.88   |
| PROIECT |  26.39   |
|   TIS   |  60.50   |
|    LS   |  85.98   |
|    CN   |  71.82   |
| PRoIECL |  33.48   |
+---------+----------+
PS D:\socket programming> █
```

-

- **The words with clear and dark handwriting are recognized with high accuracy.**

- 

THIS IS CN PROJECT

THIS IS CN PROJECT !!

- **The words which are in lower line is comparatively less accurate.**

# CONCLUSION

**Text recognition is greatly dependent the condition of handwriting the contrast, Brightness ratios in image and many more factors.**

**To detect the text very accurately our model should be more robust, and fault tolerant in any condition at the same time our model should be fast.**

**So, with the fast and accurate model, server can send data to client very fast and accurate. So, communication will be faster.**

**With this conclusion summing-up my project.**