# SMART ID SCANNER

**1 INTRODUCTION**

**1.1 Overview**

Imagine there was an easy way to get or extract text out of an image and quickly paste it into another document. The good news is that you no longer have to waste time typing everything out because there are programs that use Optical Character Recognition (OCR) to analyze the letters and words in an image, and then convert them to text.

There are a number of reasons why you might want to use the OCR function to copy text from an image.

* Paste text from a picture or screenshot into Microsoft Office or another document.
* Capture text in an error message, pop-up window or menu where text can’t be selected.
* Capture the text in a file directory (filename, file size, date modified).

Regardless of your situation, this type of functionality can be helpful, especially when you need to copy information from a file folder or screenshot of a website that typically would require you to spend a significant amount of time retyping all of the text.

Here, this proposed web app comes into picture, just enter the URL of the image and get the text. If you wish to save the text, you can save it. This web app makes your work easy

**1.2 Purpose**

If you have ever spent hours searching for content, words or other information in a collection of documents, this web app may be your new best friend. The ability to use a PDF reader, or other content management tool can be a significant time saver. For most of us in business, we are constantly looking for ways to streamline processes and increase productivity. Smart Id Scanner can be a significant enabler in this pursuit.

**Smart Id Scanner can make your life easier by:**

* Making paper-based information searchable in seconds, rather than hours.
* Reduce or eliminate costly data entry by automatically grabbing information you need from paper and putting it where it needs to go.
* Enabling entirely new ways to process documents that can eliminate “human touches”, thereby reducing costs and dramatically reducing processing times.

For example, many companies use Forms Recognition software (a type of OCR) to extract data from documents and then use that data to kick-off some number of process steps a document must go through. Examples that are common are eliminating the data entry associated with processing medical claims forms and accounts payable invoices. A favorite of ours is using a combination of OCR and AI to automatically compare an invoice to its PO, confirming that pricing and quantities are as originally quoted. The system can then flag items with discrepancies and route the document to the appropriate party for review.

**2 LITERATURE SURVEY**

**2.1 Existing Problem**

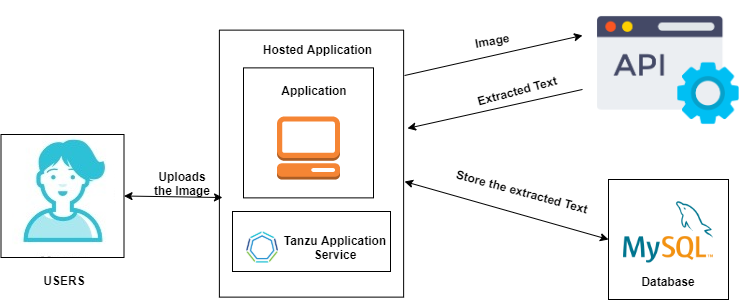
The challenge of extracting text from images of documents has traditionally been referred to as [Optical Character Recognition (OCR) and has been the focus of much research](https://static.googleusercontent.com/media/research.google.com/en/pubs/archive/33418.pdf). When documents are clearly laid out and have global structure (for example, a business letter), existing tools for OCR can perform quite well. A popular open source tool for OCR is the Tesseract Project, which was originally developed by Hewlett-Packard but has been under the care and feeding of Google in recent years. Tesseract provides an easy-to-use interface as well as an accompanying Python client library, and tends to be a go-to tool for OCR-related projects. More recently, cloud service providers are rolling out text detection capabilities alongside their various computer vision offerings. These include [Google Vision](https://cloud.google.com/vision/), [AWS Textract](https://aws.amazon.com/textract/), [Azure OCR](https://azure.microsoft.com/en-us/blog/how-to-leverage-ocr-to-full-text-search-your-images-within-azure-search/), and [Dropbox](https://blogs.dropbox.com/tech/2017/04/creating-a-modern-ocr-pipeline-using-computer-vision-and-deep-learning/), among others. It is an exciting time in the field, as computer vision techniques are becoming widely available to empower many use cases.

**2.2 Proposed Solution**

The proposed solution is a web app where we need to register first and then we need to login with those credentials. After logging in, we can enter the URL and get the text which is in the image. If we want to save the text, we can save the text. We can view the image, by clicking the view button. An api is used to extract the text from the image, which makes the whole thing very easy.

**3 THEORITICAL ANALYSIS**

**3.1 Block Diagram**



**3.3 Hardware / Software Designing**

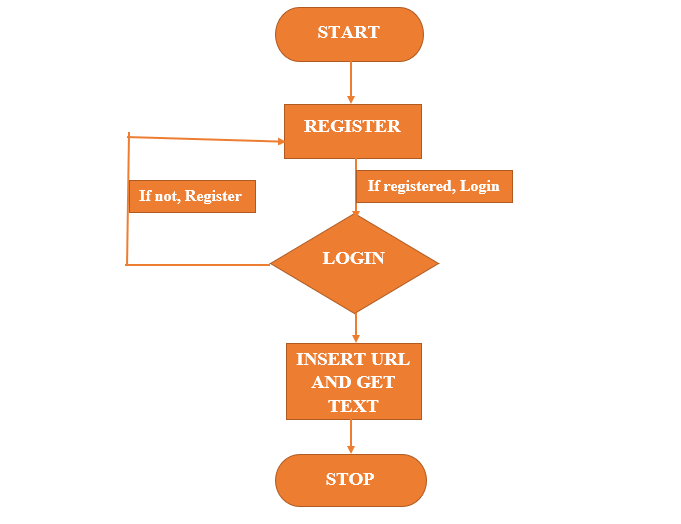
Software requirements:

* Python
* Docker

**4 EXPERIMENTAL INVESTIGATIONS**

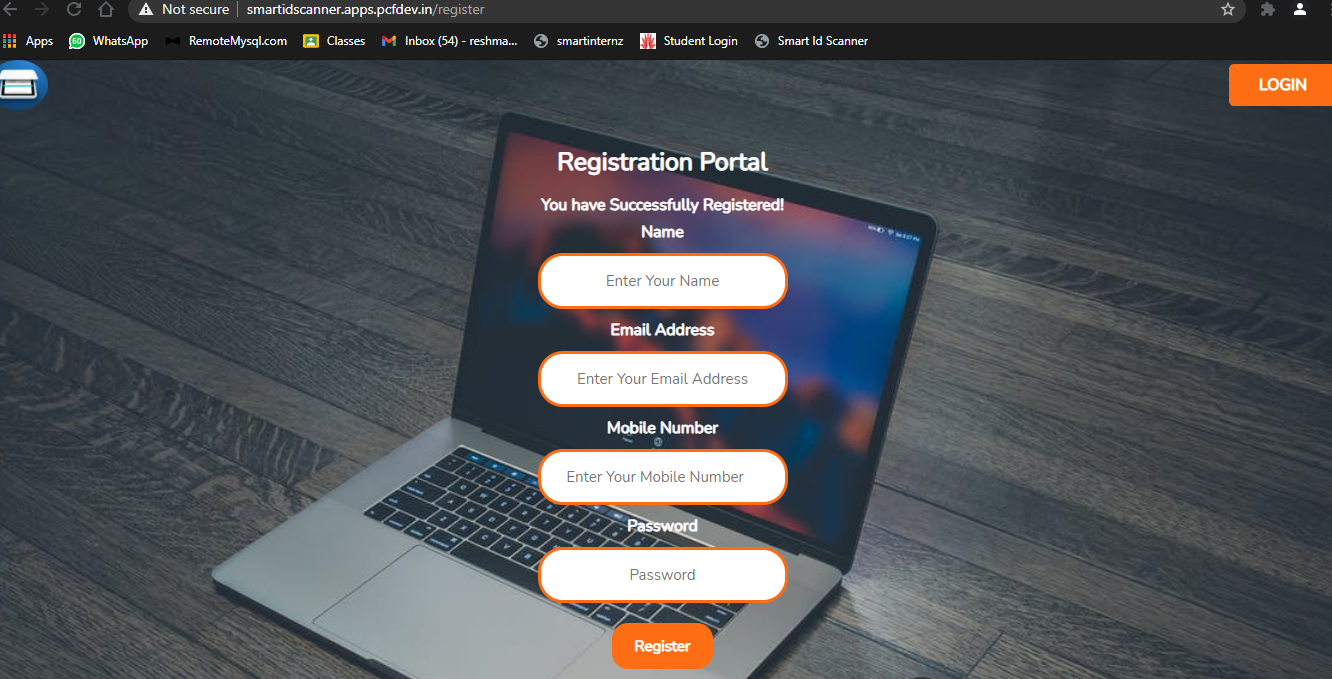
An effective methodology for text extraction from images using Gabor filter is proposed. This approach is completed by Gabor Filter, morphological and Heuristic filtering process methods is used to localize the text region better. This technique is completed by text extraction utilizing Gabor filter method which is utilized for text identification within complex images and video frames. Diverse experiments were led to assess the execution of the proposed calculation and algorithm and compare with other methods. Experimental results tested from a large dataset and demonstrated that the proposed method is effective and practical. Various parameters like a precision and recall rates are analyzed for both existing and proposed method to determine the success and limitation of our method. Experiment results show that our method can obtain 99.11 % recall rate and precision rate 94.67 % with average computational time 5.28 second /frames.

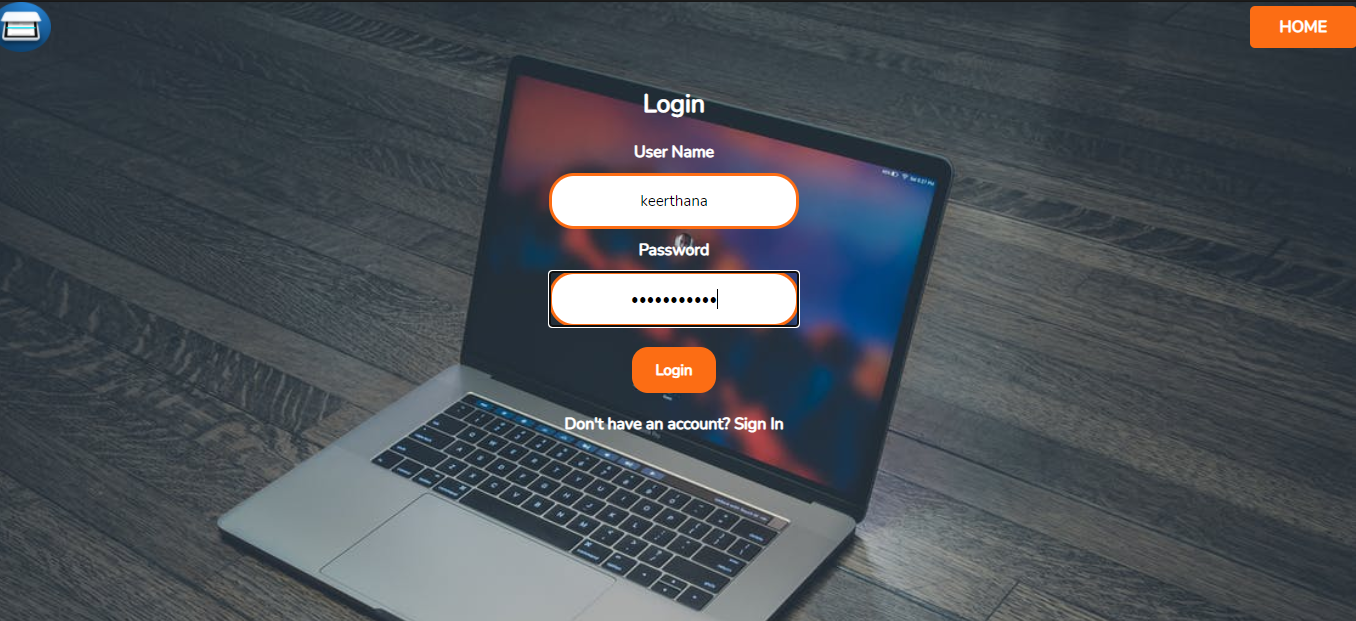
**5 FLOWCHART**

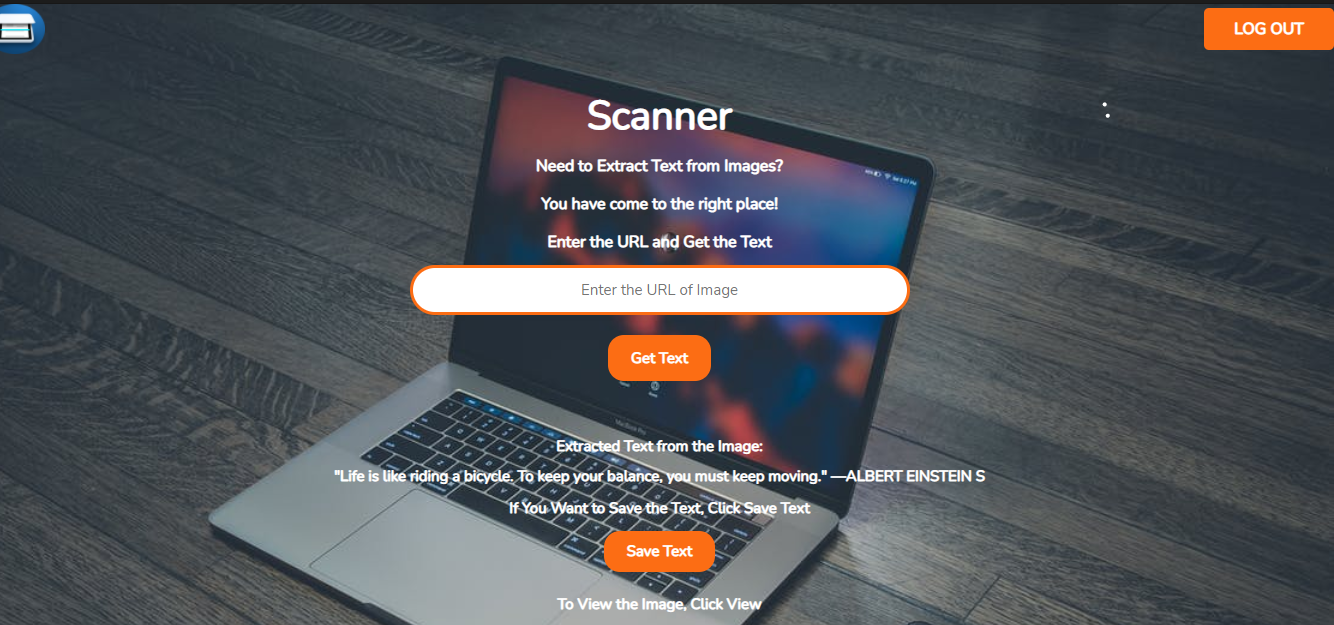


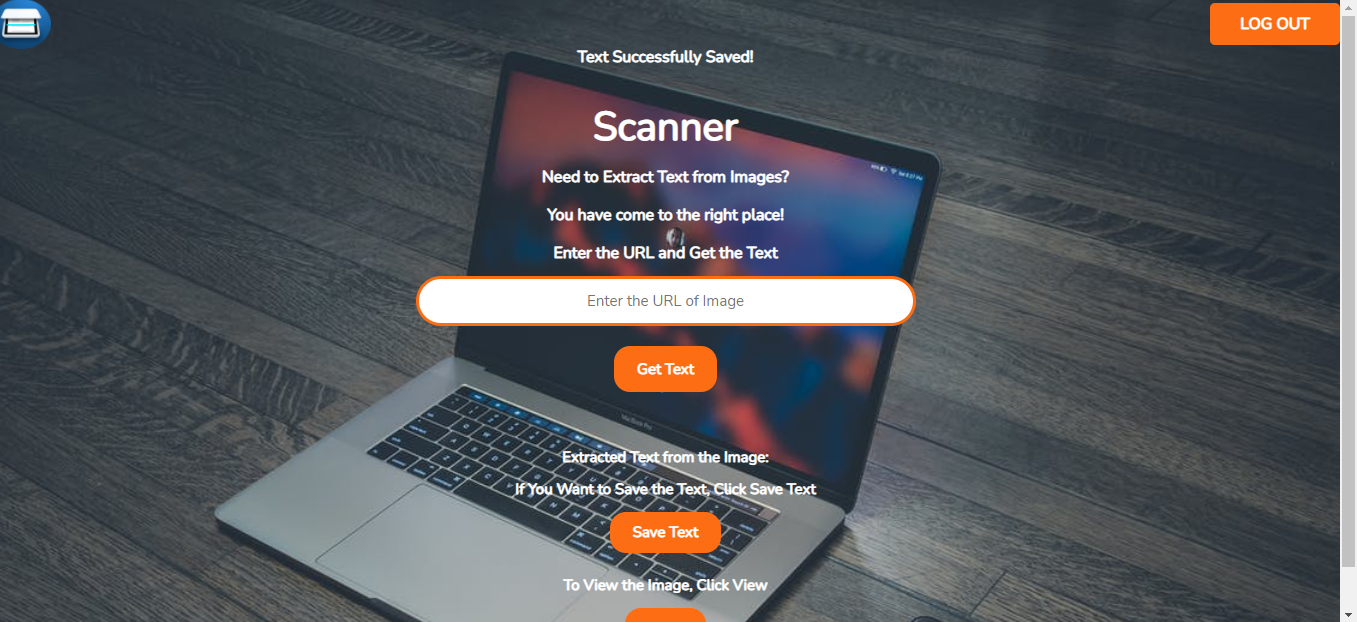
**6 RESULT**











**7 ADVANTAGES AND DISADVANTAGES**

**Advantages:**

Following are the advantages of Smart Id Scanner:

* Information provided by the Smart Id Scanner from the image can be readable with high degree of accuracy.
* Processing of Smart Id Scanner information is fast. Large quantities of text are often input quickly.
* A paper-based form is often becoming 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.
* This process is much faster as compared to the manual typing the information into the system.

**Disadvantages:**

Following are the drawbacks of Smart Id Scanner:

* This web app works efficiently with the printed text only and not with handwritten text. Handwriting must be learnt by the pc.
* There is the need of lot of space required by the image produced.
* 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**

There are numerous applications of a text information extraction system, including document analysis, vehicle license plate extraction, technical paper analysis, and object- oriented data compression. In the following, we briefly describe some of these applications.

* Wearable or portable computers: with the rapid development of computer hardware technology, wearable computers are now a reality. A TIE system involving a hand-held device and camera was presented as an application of a wearable vision system. Translation camera can detect text in a scene image and translate Japanese text into English after performing character recognition.
* Content-based video coding or document coding: The MPEG-4 standard supports object- based encoding. When text regions are segmented from other regions in an image, this can provide higher compression rates and better image quality. As a result, they can achieve a higher quality rendering of documents containing text, pictures, and graphics. Text-based image indexing: This involves automatic text-based video structuring methods using caption data.
* License/container plate recognition: There has already been a lot of work done on vehicle license plate and container plate recognition. Although container and vehicle license plates share many characteristics with scene text, many assumptions have been made regarding the image acquisition process (camera and vehicle position and direction, illumination, character types, and color) and geometric attributes of the text.

**9 CONCLUSION**

Current technologies do not work well for documents with text printed against shaded or textured backgrounds or those with non- structured layout. In contrast, our technique works well for normal documents as well as with the documents described above situations. The system is stable and robust. All the system parameters remain the same throughout all the experiments. However, it is not easy to use commercial OCR software for recognizing text extracted from images or video frames. New OCR systems need to be developed to handle large amount of noise and distortion in TIE applications.

**10 FUTURE SCOPE**

This web app does not include handwritten and complex font text which can be considered as future work.

**11 BIBILOGRAPY**

<https://www.onlineocr.net/>

<https://www.techsmith.com/blog/extract-text-from-image/>

**12 APPENDIX**

**A. SOURCE CODE**

1. import requests
2. import json
3. from flask import Flask, render\_template, request, session, redirect
4. from flask\_mysqldb import MySQL
6. app = Flask(\_\_name\_\_)
8. app.secret\_key = 'g'
10. app.config["MYSQL\_HOST"] = "remotemysql.com"
11. app.config["MYSQL\_USER"] = "l5T3J7NLMF"
12. app.config["MYSQL\_PASSWORD"] = "XU6jM5qIVw"
13. app.config["MYSQL\_DB"] = "l5T3J7NLMF"
15. mysql = MySQL(app)
17. @app.route('/home')
18. def home():
19. return render\_template('home.html')
21. @app.route('/register', methods = ["POST"])
22. def register():
23. if request.method == "POST":
24. name = request.form["name"]
25. email = request.form["email"]
26. mobileno = request.form["mobileno"]
27. password = request.form["password"]
28. cursor = mysql.connection.cursor()
29. cursor.execute('INSERT INTO registrationData VALUES (NULL,% s,% s,% s,% s)',(name,email,mobileno,password))
30. mysql.connection.commit()
31. msg = 'You have Successfully Registered!'
32. return render\_template('home.html', msg = msg)
33. @app.route('/login', methods = ["POST", "GET"])
34. def login():
35. global userid
36. global account
37. msg = ""
38. if request.method == "POST":
39. user\_name = request.form['name']
40. password = request.form['password']
41. cursor = mysql.connection.cursor()
42. cursor.execute('SELECT \* FROM registrationData WHERE name = % s AND password = % s',(user\_name, password),)
43. account = cursor.fetchone()
44. print (account)
45. if account:
46. session['loggedin'] = True
47. session['id'] = account[0]
48. userid = account[0]
49. session['email'] = account[2]
50. session['username'] = account[1]
51. msg = 'You have Sucessfully Logged In!'
52. return render\_template('scan.html', msg = msg)
53. else:
54. msg = 'Incorrect User Name / Password'
56. return render\_template('login.html', msg = msg)
58. def ocr\_space\_url(url, overlay=False, api\_key='50d04a4f1288957', language='eng'):
59. """ OCR.space API request with remote file.
60. Python3.5 - not tested on 2.7
61. :param url: Image url.
62. :param overlay: Is OCR.space overlay required in your response.
63. Defaults to False.
64. :param api\_key: OCR.space API key.
65. Defaults to 'helloworld'.
66. :param language: Language code to be used in OCR.
67. List of available language codes can be found on https://ocr.space/OCRAPI
68. Defaults to 'en'.
69. :return: Result in JSON format.
70. """
72. payload = {'url': url,
73. 'isOverlayRequired': overlay,
74. 'apikey': api\_key,
75. 'language': language,
76. }
77. r = requests.post('https://api.ocr.space/parse/image',
78. data=payload,
79. )
80. return r.content.decode()
81. @app.route('/upload', methods = ["POST"])
82. def upload\_url():
83. global output
84. if request.method == "POST":
85. url = request.form["url"]
86. name = session["username"]
87. email = session["email"]
88. cursor = mysql.connection.cursor()
89. cursor.execute('INSERT INTO saveURL VALUES(NULL,% s,% s,% s)',(name,email,url))
90. mysql.connection.commit()
91. test\_url = ocr\_space\_url(url , language='eng')
92. print(test\_url)
93. data=json.loads(test\_url)
94. output = data['ParsedResults'][0]['ParsedText']
95. session['text'] = str(output)
96. return render\_template("scan.html", output = output)
97. @app.route('/save')
98. def save\_text():
99. name = session["username"]
100. email = session["email"]
101. text = session['text']
102. new\_text = text.replace("\n"," ").replace("\r","").replace(",", "")
103. cursor = mysql.connection.cursor()
104. cursor.execute('INSERT INTO saveText VALUES(NULL,% s,% s,% s)',(name,email,new\_text))
105. mysql.connection.commit()
106. msg = 'Text Successfully Saved!'
107. return render\_template('scan.html', msg = msg)
109. @app.route('/view')
110. def view():
111. name = session["username"]
112. cursor = mysql.connection.cursor()
113. cursor.execute('SELECT \* FROM saveURL WHERE name = % s',(name,))
114. data = cursor.fetchone()
115. url = data[3]
116. return redirect(url)
117. @app.route('/logout')
118. def logout():
119. session.pop('loggedin', None)
120. session.pop('id', None)
121. session.pop('username', None)
122. session.pop('useremail', None)
123. session.pop('text', None)
124. return render\_template('home.html')
125. if \_\_name\_\_ == '\_\_main\_\_':
126. app.run(host='0.0.0.0',debug = True,port=8080**)**