MALAYSIA-JAPAN INTERNATIONAL INSTITUTE OF TECHNOLOGY
ELECTRONIC SYSTEMS ENGINEERING DEPARTMENT

SEMESTER 1 2022 /2023

SMJE 4383 ADVANCED PROGRAMMING

ASSIGNMENT 2

SCREEN SCRAPING & OCR TEXT RECOGNITION

NAME	MATRIC NO.
1. SHIM CHUNG SIONG	A19MJ0024
2. MUHAMMAD AMIRUL AFIQ BIN MOHD SHUKRI	A19MJ0145
NAME OF LECTURER	**IR. DR. ZOOL HILMI ISMAIL**

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1 Background Study

Screen scraping is a technique in computer programming where data is automatically extracted from the graphical user interface (GUI) of one application and converted into a structured format that can be used by another application. It is often used to extract data from websites or desktop applications for use in data analysis or migration to another platform. Screen scraping can be done manually or through automated scripts and software [1].

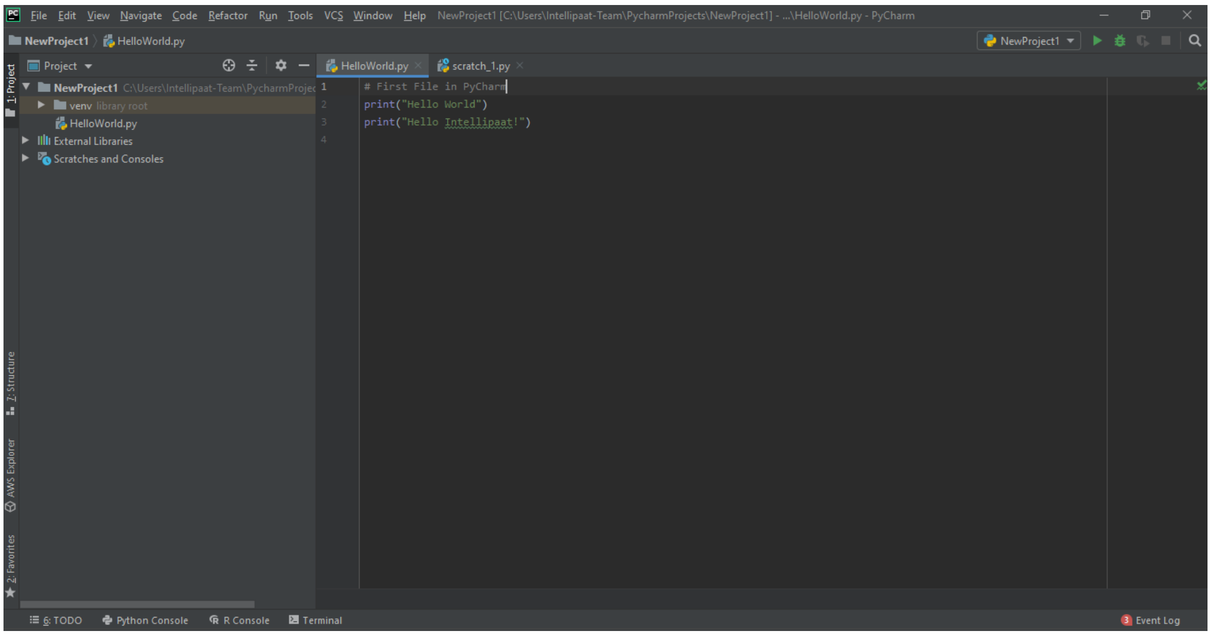
OCR (Optical Character Recognition) is a technology that enables the conversion of scanned images of text into machine-readable text. It allows you to take a picture of text, such as a scanned document or a screen shot, and extract the text content from the image into a format that can be edited and searched. OCR is commonly used for digitizing printed books and documents, converting images of text into editable text in PDFs, and recognizing text in photos and screenshots. The accuracy of OCR technology can vary, depending on the quality of the image and the complexity of the text being recognized [2].

The combination of screen scraping, and OCR text recognition refers to a process where information is automatically extracted from a graphical user interface (GUI) of an application or a website and then converted into machine-readable text using OCR technology. The screen scraping component of this process retrieves the graphical data from the source, while the OCR component converts the graphical text into editable text. This combination is useful in scenarios where a large amount of data is present in a non-machine-readable format, such as scanned documents or webpages with text embedded in images. The combination of these technologies allows the data to be automatically extracted and transformed into a usable format, saving time and effort compared to manual data entry [3].

1.2 Project Framework and Interface

In the project development, we have used several software frameworks and interfaces to complete our task. First, we have used Python 3 programming language to develop the code for OCR text recognition. Python is a high-level, interpreted, and general-purpose programming language. We have chosen Python as the project's coding language because it has rich library of pre-written code for common tasks and dynamic typing and automatic memory management.

Secondly, we have used PyCharm as the coding environment in our project. PyCharm is an Integrated Development Environment (IDE) for Python programming. It is a powerful and feature-rich platform that provides a comprehensive set of tools for coding, testing, debugging, and profiling Python applications [4].


File Edit View Navigate Code Refactor Run Tools VCS Window Help
NewProject1 [C:\Users\Intellipaat-Team\PycharmProjects\NewProject1] - ...\HelloWorld.py - PyCharm
NewProject1 HelloWorld.py
Project
NewProject1 C:\Users\Intellipaat-Team\PycharmProjec
venv library root
HelloWorld.py
External Libraries
Scratches and Consoles
HelloWorld.py
scratch_1.py
6: TODO
Python Console
R Console
Terminal
Event Log

```
# First File in PyCharm
print("Hello World")
print("Hello Intellipaat!")
```

Figure 1.1: PyCharm IDE

Lastly, the library we used for OCR text recognition in this project is Tesseract OCR is an open-source optical character recognition (OCR) engine developed by Google. It is widely used for OCR tasks and has been integrated into many OCR applications. The Tesseract library is available for use in Python through the pytesseract library, which is a Python wrapper for the Tesseract engine [5].

1.3 Working Principle

This project started with installation of PyCharm with several Python interpreter inside the IDE such as Tesseract and OpenCV. Next, by referring to some codes from GitHub and Youtube, we developed a Python code that can perform OCR text recognition. It is necessary to perform debugging procedures on the provided code in order to further analyse the text recognition such as coordinate of detection boxes and the results of text recognition.

1.4 Existing System

Text recognition has blended into our daily life long ago. Some application that we often use are implemented with text recognition. For example, Google Cloud Vision API, a cloud-based OCR service that provides fast and accurate text recognition, with support for multiple languages. Another software that has text recognition is Adobe. The extensions from Adobe such as Adobe Acrobat Pro DC and Adobe Scan allows users to recognize text in scanned documents and images [6] [7].

1.5 Problem Statement

Students need to carry out a programming task that can execute end-to-end process for Screen Scraping & OCR Text Recognition using Python Script. This topic is selected because it is part of the academia-industrial collaboration at MJIIT.

1.6 Project Objective

1. To extract the information of text recognition and display the data to user

2. To automate the process of text recognition without excessive action taken by user

CHAPTER 2

METHODOLOGY

2.1 Software Required

1. PyCharm
2. Python 3

2.2 Library Used

1. Tesseract
2. OpenCV
3. NumPy

2.3 Project Flow

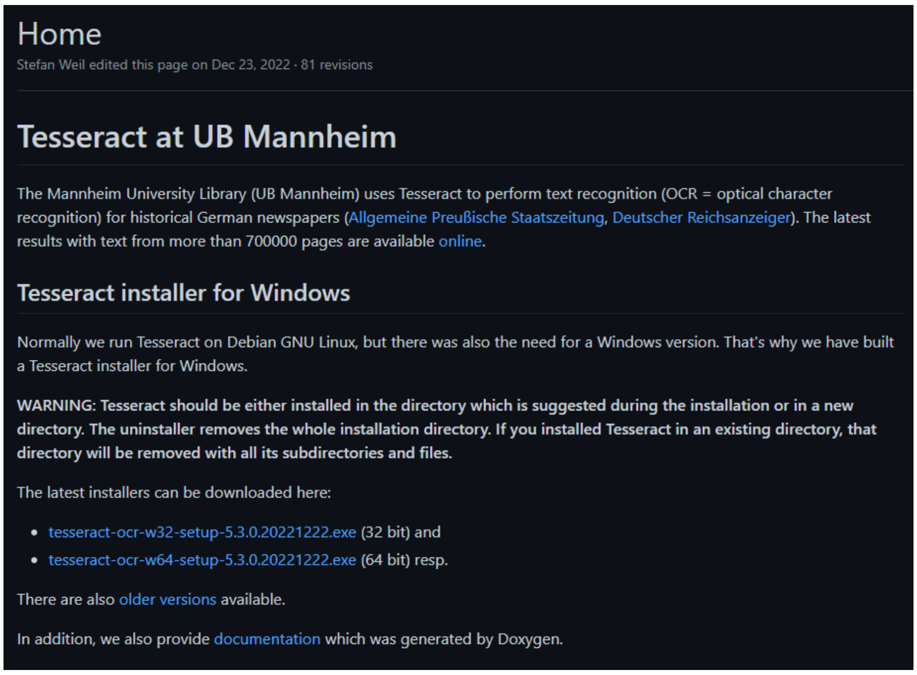
The main objective of this project is to detect the text and number in an image and show the result of detection on the boxes of detection.

2.3.1 Library Preparation

Tesseract library

The tesseract library package is downloaded manually from the GitHub website.

Website URL: https://github.com/UB-Mannheim/tesseract/wiki



Home

Stefan Weil edited this page on Dec 23, 2022 · 81 revisions

Tesseract at UB Mannheim

The Mannheim University Library (UB Mannheim) uses Tesseract to perform text recognition (OCR = optical character recognition) for historical German newspapers (Allgemeine Preußische Staatszeitung, Deutscher Reichsanzeiger). The latest results with text from more than 700000 pages are available online.

Tesseract installer for Windows

Normally we run Tesseract on Debian GNU Linux, but there was also the need for a Windows version. That's why we have built a Tesseract installer for Windows.

WARNING: Tesseract should be either installed in the directory which is suggested during the installation or in a new directory. The uninstaller removes the whole installation directory. If you installed Tesseract in an existing directory, that directory will be removed with all its subdirectories and files.

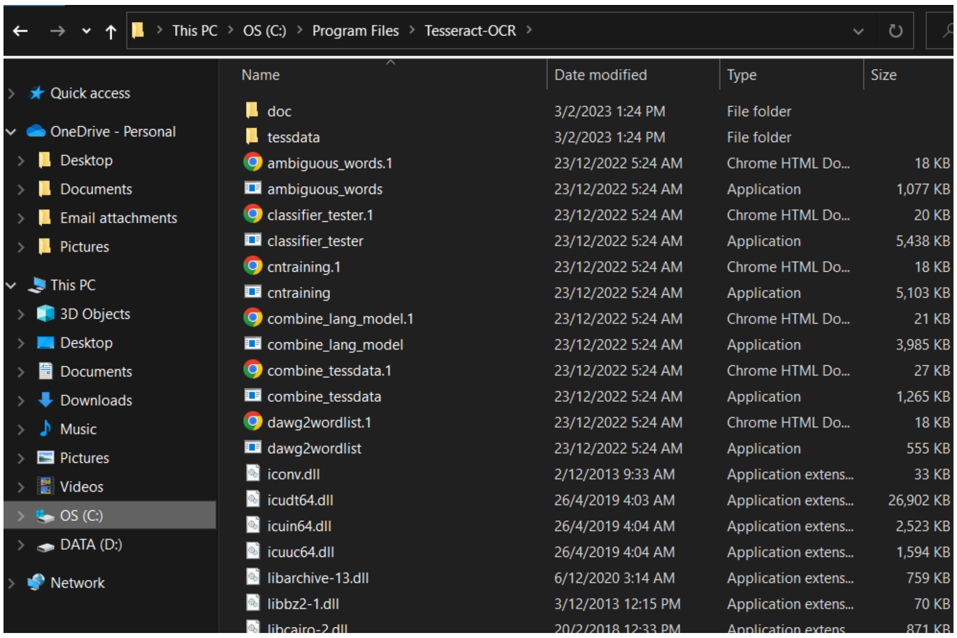
The latest installers can be downloaded here:

- tesseract-ocr-w32-setup-5.3.0.20221222.exe (32 bit) and
- tesseract-ocr-w64-setup-5.3.0.20221222.exe (64 bit) resp.

There are also older versions available.

In addition, we also provide documentation which was generated by Doxygen.

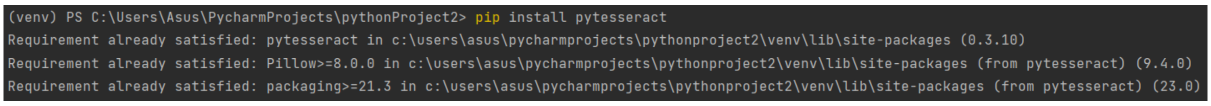
Figure 2.1: GitHub website for tesseract library



Name	Date modified	Type	Size
doc	3/2/2023 1:24 PM	File folder	
tessdata	3/2/2023 1:24 PM	File folder	
ambiguous_words.1	23/12/2022 5:24 AM	Chrome HTML Do...	18 KB
ambiguous_words	23/12/2022 5:24 AM	Application	1,077 KB
classifier_tester.1	23/12/2022 5:24 AM	Chrome HTML Do...	20 KB
classifier_tester	23/12/2022 5:24 AM	Application	5,438 KB
cntraining.1	23/12/2022 5:24 AM	Chrome HTML Do...	18 KB
cntraining	23/12/2022 5:24 AM	Application	5,103 KB
combine_lang_model.1	23/12/2022 5:24 AM	Chrome HTML Do...	21 KB
combine_lang_model	23/12/2022 5:24 AM	Application	3,985 KB
combine_tessdata.1	23/12/2022 5:24 AM	Chrome HTML Do...	27 KB
combine_tessdata	23/12/2022 5:24 AM	Application	1,265 KB
dawg2wordlist.1	23/12/2022 5:24 AM	Chrome HTML Do...	18 KB
dawg2wordlist	23/12/2022 5:24 AM	Application	555 KB
iconv.dll	2/12/2013 9:33 AM	Application extens...	33 KB
icudt64.dll	26/4/2019 4:03 AM	Application extens...	26,902 KB
icuin64.dll	26/4/2019 4:04 AM	Application extens...	2,523 KB
icuuc64.dll	26/4/2019 4:04 AM	Application extens...	1,594 KB
libarchive-13.dll	6/12/2020 3:14 AM	Application extens...	759 KB
libbz2-1.dll	3/12/2013 12:15 PM	Application extens...	70 KB
libcairo-2.dll	20/2/2018 12:33 PM	Application extens...	871 KB

Figure 2.2: Tesseract file in window

After the tesseract is installed into window, we install the package using the terminal inside PyCharm by using pip install command.

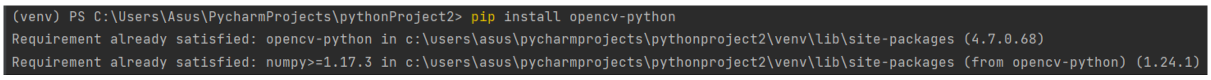


```
(venv) PS C:\Users\Asus\PycharmProjects\pythonProject2> pip install pytesseract
Requirement already satisfied: pytesseract in c:\users\asus\pycharmprojects\pythonproject2\venv\lib\site-packages (0.3.10)
Requirement already satisfied: Pillow>=8.0.0 in c:\users\asus\pycharmprojects\pythonproject2\venv\lib\site-packages (from pytesseract) (9.4.0)
Requirement already satisfied: packaging>=21.3 in c:\users\asus\pycharmprojects\pythonproject2\venv\lib\site-packages (from pytesseract) (23.0)
```

Figure 2.3: pip install command for tesseract installation

OpenCV library

OpenCV library is also installed by using same method with tesseract library. Pip install command is run in PyCharm terminal.

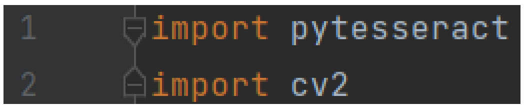


```
(venv) PS C:\Users\Asus\PycharmProjects\pythonProject2> pip install opencv-python
Requirement already satisfied: opencv-python in c:\users\asus\pycharmprojects\pythonproject2\venv\lib\site-packages (4.7.0.68)
Requirement already satisfied: numpy>=1.17.3 in c:\users\asus\pycharmprojects\pythonproject2\venv\lib\site-packages (from opencv-python) (1.24.1)
```

Figure 2.4: pip install command for OpenCV installation

2.3.2 Codes

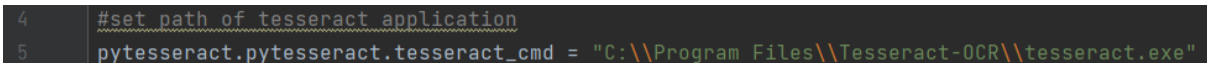
In the first 2 lines, we have imported the libraries that will be used for OCR text recognition.



```
1  import pytesseract
2  import cv2
```

Figure 2.5: Coding of line 1 and 2

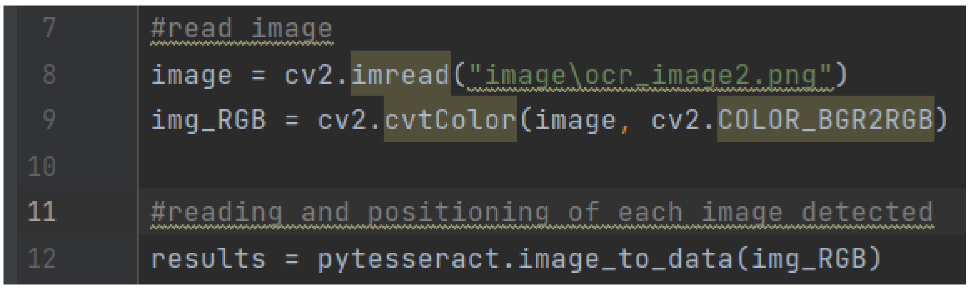
Next, we set the path of tesseract executable file to the tesseract command in the codes.



```
4  #set path of tesseract application
5  pytesseract.pytesseract.tesseract_cmd = "C:\\Program Files\\Tesseract-OCR\\tesseract.exe"
```

Figure 2.6: Coding of line 4 and 5

After the setup is complete, we read the image and store the results into a variable.

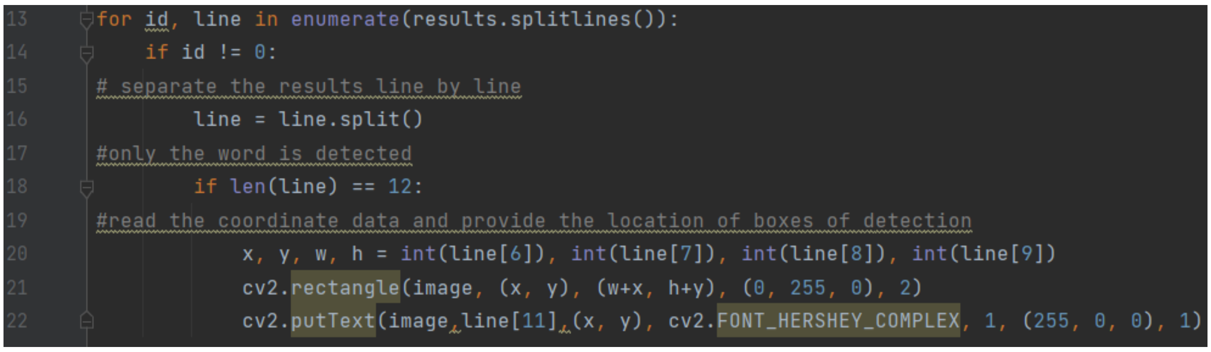


```
7   #read image
8   image = cv2.imread("image\ocr_image2.png")
9   img_RGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
10
11  #reading and positioning of each image detected
12  results = pytesseract.image_to_data(img_RGB)
```

Figure 2.7: Coding of line 7 to 12

Figure 2.8: Input image

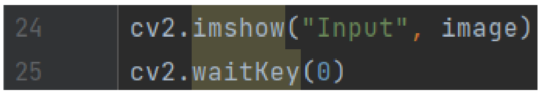
The results is further analyzed by splitting the line and store the coordinates to the variables.



```python
13 for id, line in enumerate(results.splitlines()):
14     if id != 0:
15 # separate the results line by line
16         line = line.split()
17 #only the word is detected
18         if len(line) == 12:
19 #read the coordinate data and provide the location of boxes of detection
20             x, y, w, h = int(line[6]), int(line[7]), int(line[8]), int(line[9])
21             cv2.rectangle(image, (x, y), (w+x, h+y), (0, 255, 0), 2)
22             cv2.putText(image,line[11],(x, y), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 0), 1)
```

Figure 2.9: Coding of line 13 to 22

Lastly, the final picture of text recognition will be displayed



```python
24 cv2.imshow("Input", image)
25 cv2.waitKey(0)
```

Figure 2.10: Coding of line 24 and 25

CHAPTER 3

RESULT AND DISCUSSION

3.1 Result

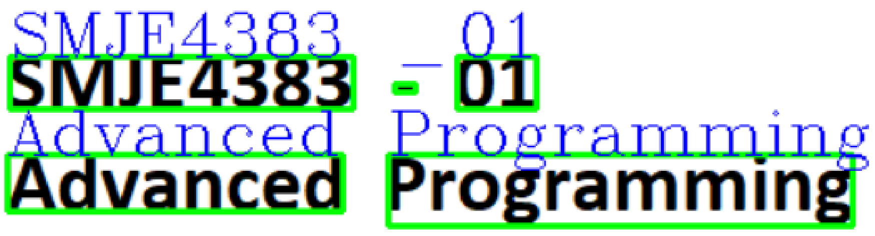
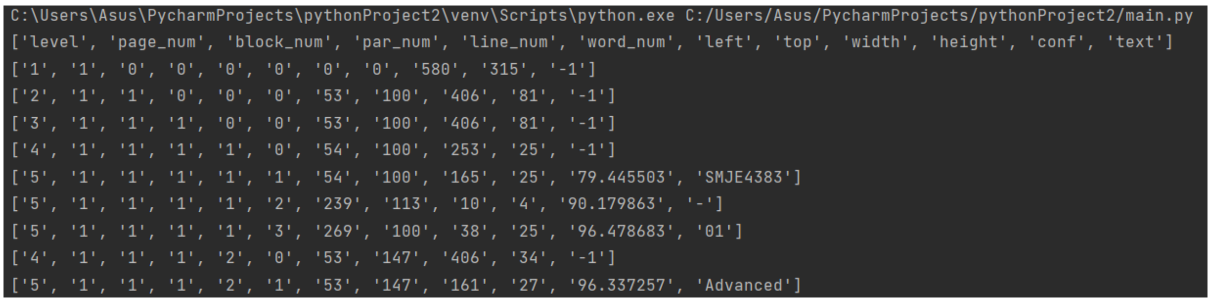


Figure 3.1: Result of the project

3.2 Discussion

In the end of our simulation, we have successfully implemented the OCR text recognition for the image. As the result shown, it can detect the words, symbol and numbers and print the result of recognition on it. In order to provide correct location for detection boxes and recognition result, we printed out the coordinate data of each text detection such as left, top, width and height.



```
C:\Users\Asus\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:/Users/Asus/PycharmProjects/pythonProject2/main.py
['level', 'page_num', 'block_num', 'par_num', 'line_num', 'word_num', 'left', 'top', 'width', 'height', 'conf', 'text']
['1', '1', '0', '0', '0', '0', '0', '0', '580', '315', '-1']
['2', '1', '1', '0', '0', '0', '53', '100', '406', '81', '-1']
['3', '1', '1', '1', '0', '0', '53', '100', '406', '81', '-1']
['4', '1', '1', '1', '1', '0', '54', '100', '253', '25', '-1']
['5', '1', '1', '1', '1', '1', '54', '100', '165', '25', '79.445503', 'SMJE4383']
['5', '1', '1', '1', '1', '2', '239', '113', '10', '4', '90.179863', '-']
['5', '1', '1', '1', '1', '3', '269', '100', '38', '25', '96.478683', '01']
['4', '1', '1', '1', '2', '0', '53', '147', '406', '34', '-1']
['5', '1', '1', '1', '2', '1', '53', '147', '161', '27', '96.337257', 'Advanced']
```

Figure 3.2: Coordinate of text recognition

CHAPTER 4

CONCLUSION

In conclusion, we have fulfilled the objective of this project hence solved the problem stated in problem statement. Further study can be conducted to develop a more advanced OCR text recognition in future.

Reference

[1] Selenium with Python (n.d.). Selenium with Python — Selenium Python bindings for automating web browsers. Retrieved from: https://selenium-python.readthedocs.io/

[2] Tesseract OCR (n.d.). Tesseract Open Source OCR Engine (main repository). Retrieved from: https://github.com/tesseract-ocr/tesseract

[3] Scrapy-OCR (n.d.). Scrapy-OCR | An OCR engine based on Scrapy. Retrieved from: https://github.com/scrapy-plugins/scrapy-ocr

[4] PyCharm (n.d.). PyCharm: The Python IDE for Professional Developers. Retrieved from: https://www.jetbrains.com/pycharm/

[5] Tesseract OCR (n.d.). Tesseract OCR. Retrieved from: https://github.com/tesseract-ocr/tesseract

[6] Adobe Acrobat Pro DC (n.d.). Adobe Acrobat Pro DC: PDF converter, convert PDFs from anywhere. Retrieved from: https://acrobat.adobe.com/us/en/acrobat/pro.html

[7] Adobe Scan (n.d.). Adobe Scan: PDF Scanner with OCR, PDF Creator. Retrieved from: https://acrobat.adobe.com/us/en/mobile/scan.html