UTM
UNIVERSITI TEKNOLOGI MALAYSIA
Malaysia-Japan International Institute of Technology (MJIIT)

SEMESTER 1 SESSION 2022/2023

SMJE4383 – ADVANCE PROGRAMMING

ASSIGNMENT 2

SCREEN SCRAPING & OCR TEXT RECOGNITION USING PYTHON SCRIPT

NAME	MATRIC NO.
MUHAMMAD HANIF BIN ABU KASSIM	A20MJ0060
SITI NUR IZZATI BINTI SANOSI	A19MJ3063

Section: 01

Lecturer Name: Assoc. Prof. Ir. Dr. Zool Hilmi Bin Ismail

Submission Date: 7 February 2023

Github link: https://github.com/IzzatiSanosi/Advanced-Programming/tree/Assignment1B

Introduction

Background study

Screen scraping and OCR text recognition are two important techniques in the field of data extraction and information retrieval. Screen scraping is the process of automatically extracting information from websites, while OCR (Optical Character Recognition) is the process of recognizing text from images or scanned documents. These techniques are widely used in various industries, including finance, e-commerce, and digital marketing, to gather and analyze data from a variety of sources.

The main objective of this assignment is to carry out a programming task that can execute an end-to-end process for screen scraping and OCR text recognition using Python script. This topic was selected as part of an academic-industrial collaboration at MJIIT, with the aim of solving industrial-based problems.

To gain a comprehensive understanding of this assignment, it is necessary to understand the elements and techniques involved in screen scraping and OCR text recognition. These elements include web scraping libraries such as BeautifulSoup, OCR libraries such as pytesseract, programming languages such as Python, and also can use UI.Vision RPA

This report will provide an overview of screen scraping and OCR text recognition, including their uses and benefits, and will also provide a detailed explanation of the programming task and the code used to carry out the end-to-end process. Additionally, the report will discuss the challenges and limitations of these techniques, as well as their future potential.

Problem staement

A company wants to automate their data extraction process from supplier invoices. The invoices are sent in the form of image files and the company wants to extract the following information: Invoice number, date, supplier name, items purchased, and total amount. Currently, the data extraction is being done manually by the employees which is time-consuming and prone to errors.

Objective

The objective of the assignment is to develop a solution using Screen Scraping and OCR Text Recognition in Python to extract the required information from the image invoices automatically.

Methodology

This solution can be achieved by carrying out the following steps:

1. Screen Scraping: Use a suitable library like ComputerVision 'cv2 to extract the text information from the image file.
2. OCR Text Recognition: Use an OCR library like 'pytesseract' to recognize the text in the image and convert it into a readable format.
3. Data extraction: Use regular expressions or string manipulation techniques to extract the required information from the recognized text.
4. Data validation: Validate the extracted information to ensure that it is accurate and meets the required standards.
5. Storing: Save the data systematically like in CSV file format to make it easy to update and analyze.

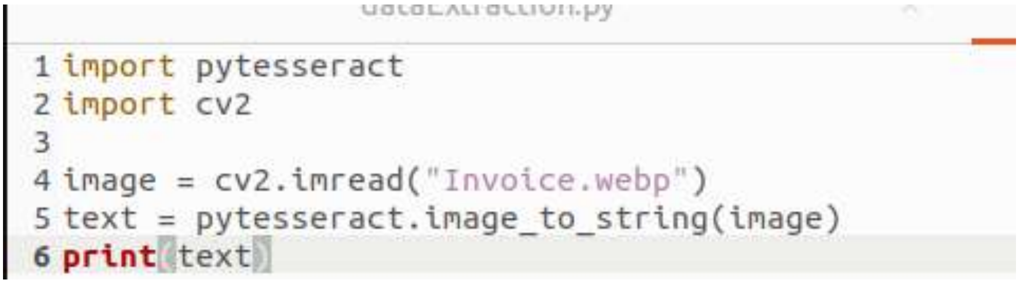
By carrying out this programming task, the company can save a significant amount of time and reduce errors in their data extraction process. The source code is available in Appendix (A).

Result and Discussin

Screen scraping and OCR text recognition

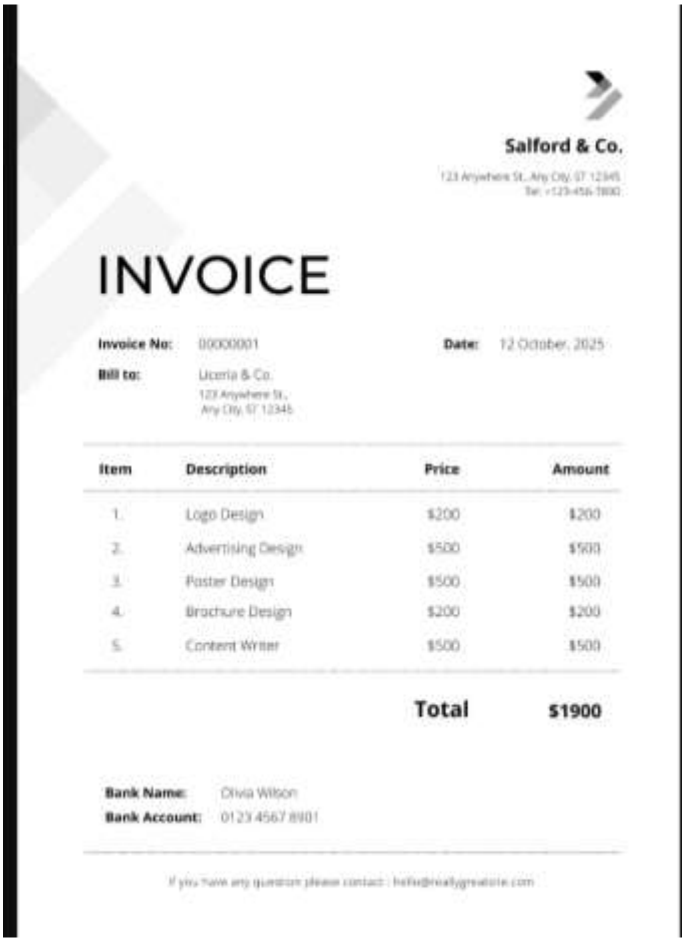
The cv2 library from the Computer Vision (CV) module will be used in the code because it is used to read an image file and perform image processing tasks. In this code, the cv2.imread() function is used to load the invoice image file, which will later be processed using Optical Character Recognition (OCR) to extract text information. The CV module provides a wide range of image processing capabilities, including image loading and manipulation, which are essential for tasks like OCR.

The ‘pytesseract’ library is used to perform Optical Character Recognition (OCR) on an image. It allows you to extract text from an image, which in this case is the invoice image, and convert it into a string of characters that can be processed and analyzed by the program. This is useful in this code because the information on the invoice is contained in an image, and in order to extract that information and process it, we need to first perform OCR on the image to convert it into a string that can be manipulated.



```python
import pytesseract
import cv2

image = cv2.imread("Invoice.webp")
text = pytesseract.image_to_string(image)
print(text)
```

Figure 1: This show how we can screen scaping and OCR text recognition of the invoice image.



Salford & Co.
123 Anywhere St., Any City, ST 12345
Tel: +123-456-7890

INVOICE

Invoice No: 00000001 **Date:** 12 October, 2025

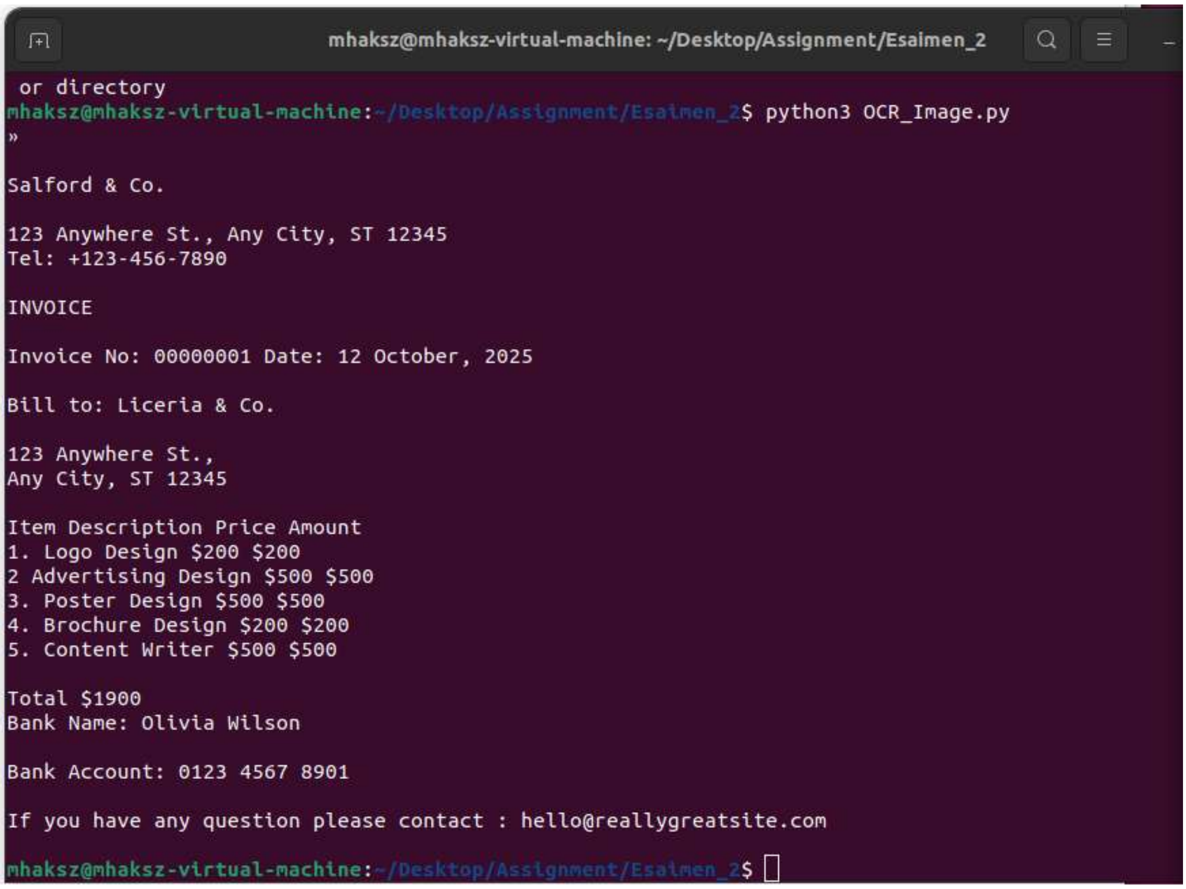
Bill to: Liceria & Co.
123 Anywhere St.,
Any City, ST 12345

Item	Description	Price	Amount
1.	Logo Design	$200	$200
2.	Advertising Design	$500	$500
3.	Poster Design	$500	$500
4.	Brochure Design	$200	$200
5.	Content Writer	$500	$500
		Total	**$1900**

Bank Name: Olivia Wilson
Bank Account: 0123 4567 8901

If you have any question please contact : hello@reallygreatsite.com

Figure 2 : Invoice image



mhaksz@mhaksz-virtual-machine: ~/Desktop/Assignment/Esaimen_2

```
 or directory
mhaksz@mhaksz-virtual-machine:~/Desktop/Assignment/Esaimen_2$ python3 OCR_Image.py
»

Salford & Co.

123 Anywhere St., Any City, ST 12345
Tel: +123-456-7890

INVOICE

Invoice No: 00000001 Date: 12 October, 2025

Bill to: Liceria & Co.

123 Anywhere St.,
Any City, ST 12345

Item Description Price Amount
1. Logo Design $200 $200
2 Advertising Design $500 $500
3. Poster Design $500 $500
4. Brochure Design $200 $200
5. Content Writer $500 $500

Total $1900
Bank Name: Olivia Wilson

Bank Account: 0123 4567 8901

If you have any question please contact : hello@reallygreatsite.com

mhaksz@mhaksz-virtual-machine:~/Desktop/Assignment/Esaimen_2$ 
```

Figure 3: Text data string after OCR text recognition

Data extraction and Validation

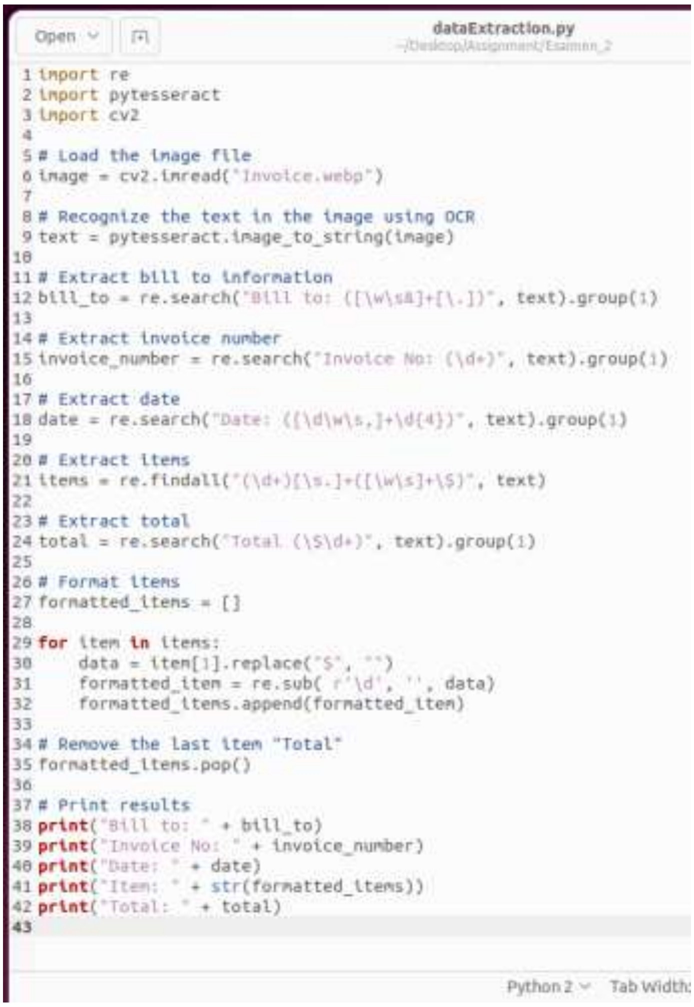
The regular expressions (re) module is used to search the string "text" obtained from the OCR to extract the desired information. In our objective, we want extract the data for bill to, invoice number, data, item, total amount. This is how we extract all of desired output based on the text data string that we obtain fromOCR text recognition.

- Bill To: The regular expression pattern "Bill to: ([\w\s&]+[\.])" is used to extract the information of the person or company who is the recipient of the bill. The pattern searches for the text "Bill to:" followed by one or more words, spaces, & symbol, and a period. The information in the parenthesis ([\w\s&]+[\.]) is extracted and stored as the value of bill_to.
- invoice_number: The regular expression pattern "Invoice No: (\d+)" is used to extract the invoice number. The pattern searches for the text "Invoice No:" followed by one or

more digits. The digits in the parenthesis (\d+) are extracted and stored as the value of invoice_number.

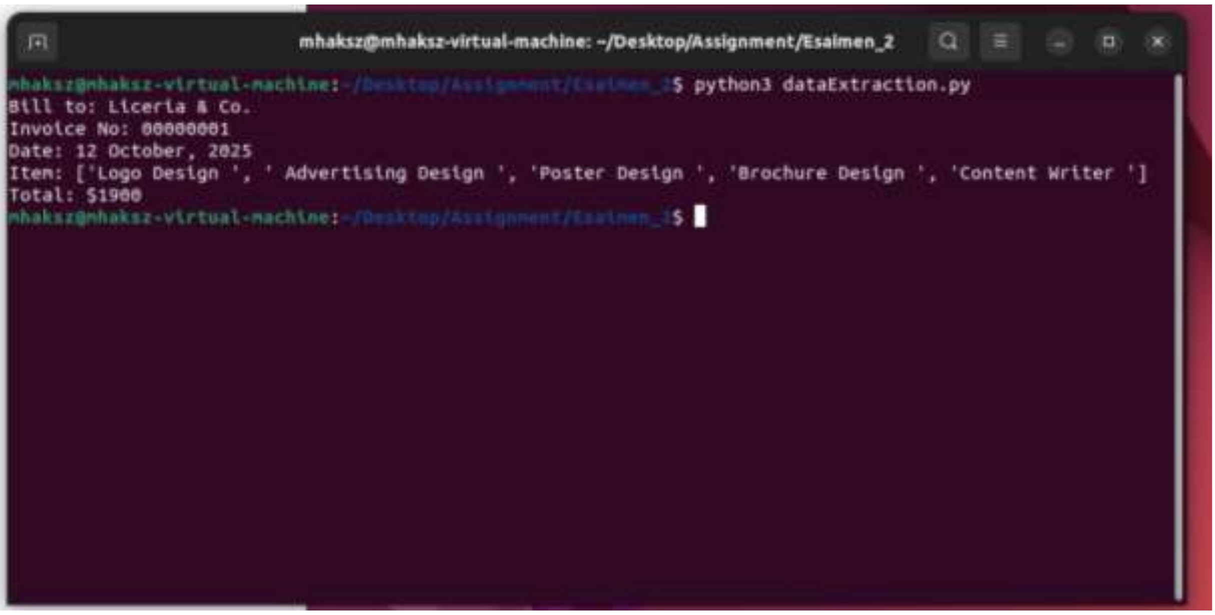
- date: The regular expression pattern "Date: ([\d\w\s,]+\d{4})" is used to extract the date information. The pattern searches for the text "Date:" followed by one or more digits, letters, spaces, or commas and ends with four digits. The information in the parenthesis ([\d\w\s,]+\d{4}) is extracted and stored as the value of date.
- items: The regular expression pattern "(\d+)[\s.]+([\w\s]+\$)" is used to extract the item information. The pattern searches for one or more digits followed by spaces, dots, and one or more words, spaces, and the "$" symbol. The digits and words in the parenthesis (\d+)[\s.]+([\w\s]+\$) are extracted and stored as a list of tuples items.
- total: The regular expression pattern "Total (\$\d+)" is used to extract the total amount information. The pattern searches for the text "Total" followed by a space and one or more digits with a "$" symbol. The amount in the parenthesis (\$\d+) is extracted and stored as the value of total.
- formatted_items: The loop for item in items: is used to format the items information by removing the "$" symbol and the digits. The loop iterates over each tuple in the items list, extracts the second element of each tuple (the item description), replaces the "$" symbol with an empty string, and removes the digits using the re.sub() function with the pattern r'\d' and an empty string as the replacement. The formatted item descriptions are then added to the list formatted_items.
- Finally, the last line formatted_items.pop() removes the last item from the formatted_items list, which is "Total".

To validate the data extraction, we print for all the data that we want extract.



```python
import re
import pytesseract
import cv2

# Load the image file
image = cv2.imread("Invoice.webp")

# Recognize the text in the image using OCR
text = pytesseract.image_to_string(image)

# Extract bill to information
bill_to = re.search("Bill to: ([\w\s&]+[\.])", text).group(1)

# Extract invoice number
invoice_number = re.search("Invoice No: (\d+)", text).group(1)

# Extract date
date = re.search("Date: ([\d\w\s,]+\d{4})", text).group(1)

# Extract items
items = re.findall("(\d+)[\s.]+([\w\s]+\$)", text)

# Extract total
total = re.search("Total (\$\d+)", text).group(1)

# Format items
formatted_items = []

for item in items:
    data = item[1].replace("$", "")
    formatted_item = re.sub( r'\d', '', data)
    formatted_items.append(formatted_item)

# Remove the last item "Total"
formatted_items.pop()

# Print results
print("Bill to: " + bill_to)
print("Invoice No: " + invoice_number)
print("Date: " + date)
print("Item: " + str(formatted_items))
print("Total: " + total)
```

Figure 4 : Data extraction and validation code

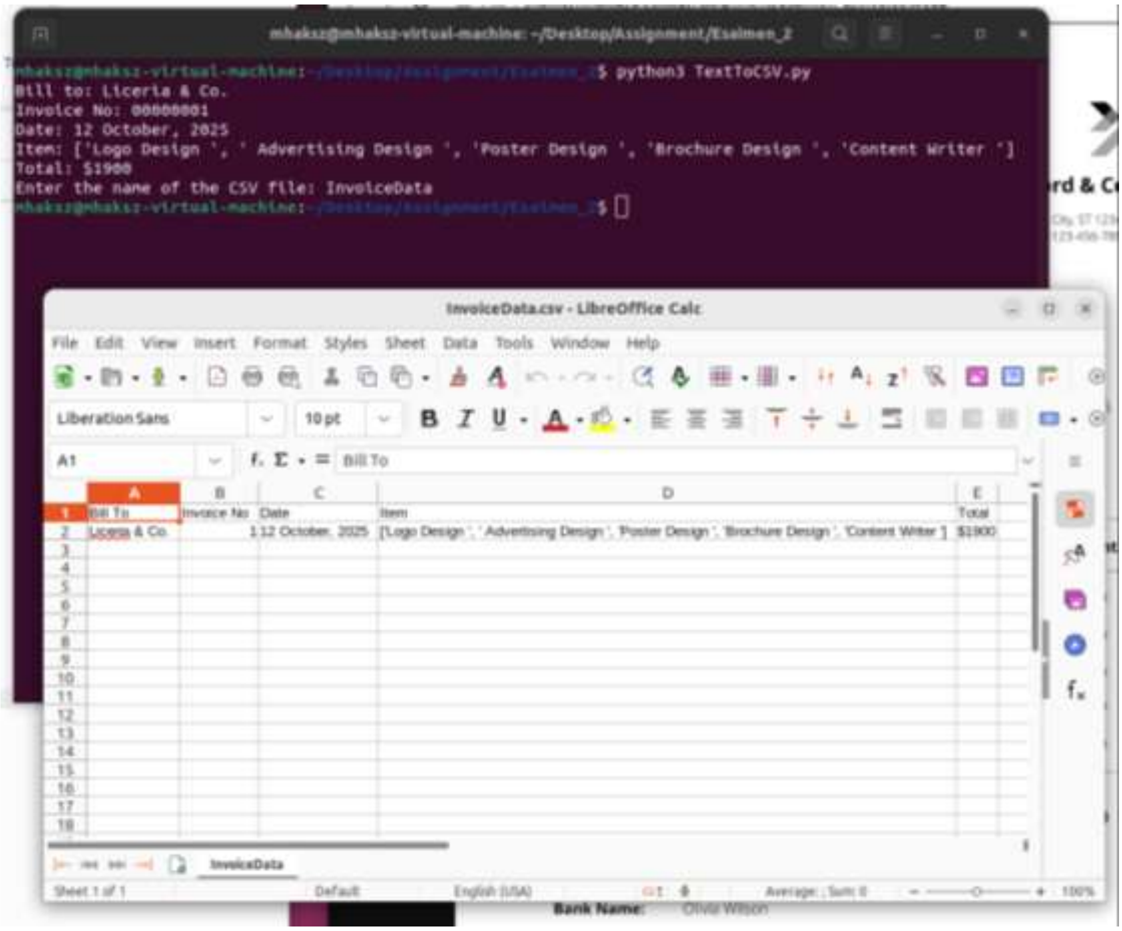


mhaksz@mhaksz-virtual-machine: ~/Desktop/Assignment/Esaimen_2

```
mhaksz@mhaksz-virtual-machine:~/Desktop/Assignment/Esaimen_2$ python3 dataExtraction.py
Bill to: Liceria & Co.
Invoice No: 00000001
Date: 12 October, 2025
Item: ['Logo Design ', ' Advertising Design ', 'Poster Design ', 'Brochure Design ', 'Content Writer ']
Total: $1900
mhaksz@mhaksz-virtual-machine:~/Desktop/Assignment/Esaimen_2$ 
```

Figure 5 : Result for data extraction and validation

Storing



```
mhaksz@mhaksz-virtual-machine:~/Desktop/Assignment/Esalmen_2$ python3 TextToCSV.py
Bill to: Liceria & Co.
Invoice No: 00000001
Date: 12 October, 2025
Item: ['Logo Design ', ' Advertising Design ', 'Poster Design ', 'Brochure Design ', 'Content Writer ']
Total: $1900
Enter the name of the CSV file: InvoiceData
mhaksz@mhaksz-virtual-machine:~/Desktop/Assignment/Esalmen_2$
```

InvoiceData.csv - LibreOffice Calc

	A	B	C	D	E
1	Bill To	Invoice No	Date	Item	Total
2	Liceria & Co.	1	12 October, 2025	['Logo Design ', ' Advertising Design ', 'Poster Design ', 'Brochure Design ', 'Content Writer ']	$1900

Figure 6 : Result for convert the text file to CSV file format

CSV (Comma Separated Values) is a simple file format for storing tabular data, where data is represented as a series of rows, each with a number of fields separated by commas. Converting the text file to CSV format makes it easier to update and analyze the data for several reasons:

1. Consistency: The use of a standard file format like CSV ensures that the data is stored in a consistent and standardized manner, making it easier to work with and understand.
2. Accessibility: CSV files are supported by a wide range of applications, including spreadsheets, databases, and data analysis tools. This makes it easy to access, manipulate, and analyze the data, even if you don't have specialized software.
3. Readability: CSV files are human-readable, which means that you can quickly inspect and understand the data, even if you don't have any programming skills.

4. Portability: CSV files are text-based, which makes them easy to transport and share. They can be easily exported from one application and imported into another, regardless of platform or operating system.

Conclusion

In conclusion, this report presented an end-to-end process for screen scraping and OCR text recognition using Python script to automate data extraction from supplier invoices for a company. The solution involved using the cv2 library from the Computer Vision (CV) module to read the invoice image and the 'pytesseract' library for OCR. The regular expressions (re) module was used to extract the required information from the recognized text and validate it. The program was able to successfully extract the information of the bill recipient, invoice number, date, items purchased, and total amount. The source code is available in appendix (A). This solution can save the company a significant amount of time and reduce errors in the data extraction process.

Appendix (A)

********Source Code********

```
import re
import csv
import pytesseract
import cv2

# Load the image file
image = cv2.imread("Invoice.webp")

# Recognize the text in the image using OCR
text = pytesseract.image_to_string(image)

# Extract bill to information
bill_to = re.search("Bill to: ([\w\s&]+[\.])", text).group(1)

# Extract invoice number
invoice_number = re.search("Invoice No: (\d+)", text).group(1)

# Extract date
date = re.search("Date: ([\d\w\s,]+\d{4})", text).group(1)

# Extract items
items = re.findall("(\d+)[\s.]+([\w\s]+\$)", text)

# Extract total
total = re.search("Total (\$\d+)", text).group(1)

# Format items
formatted_items = []

for item in items:
    data = item[1].replace("$", "")
    formatted_item = re.sub( r'\d', '', data)
    formatted_items.append(formatted_item)

# Remove the last item "Total"
formatted_items.pop()

# Print results
print("Bill to: " + bill_to)
print("Invoice No: " + invoice_number)
print("Date: " + date)
print("Item: " + str(formatted_items))
print("Total: " + total)
```

```
# Write the extracted data to a CSV file
file_name = input("Enter the name of the CSV file: ")

with open(f'{file_name}.csv', 'w', newline='') as file:
    writer = csv.writer(file)

    # Write header row
    writer.writerow(["Bill To", "Invoice No", "Date", "Item", "Total"])

    # Write data row
    writer.writerow([bill_to, invoice_number, date, formatted_items, total])
```