A Weekly Report

on

INVOICE MANAGEMENT

at

OneOnic Solution

by

Mr. Tank Viraj Rupeshbhai

EC034, 21ECUBG069

B.Tech. (EC) Semester - VIII

Faculty Supervisor

Dr. NarendraKumar Chauhan

External Guide

Mr. Umesh Ghediya

In Partial Fulfillment of Requirement of

Bachelor of Technology - Electronics & Communication

Submitted To

Department of Electronics & Communication Engineering

Faculty of Technology

Dharmsinh Desai University, Nadiad - 387001.

(Jan 2025)

Task 7 :- Invoice Text Extraction with Google Sheets Integration

Introduction :-

In this task, we will learn how to automate the process of extracting text from invoices which can be in PDF or image format and saving that text into Google Sheets. We will use Tesseract OCR to read the text from the invoices and then save the extracted text into a Google Sheet. This will help us manage and organize invoice data more efficiently.

Tool's and technology Used :-

1) Tesseract OCR:
   A popular, free OCR tool that helps computers read text from images. It is easy to use and works well with clear, printed text.
2) pytesseract:
   A Python wrapper for Tesseract OCR.
3) Google Sheets API:
   Allows us to connect to Google Sheets and save data directly into it.
4) gspread:
   Helps us interact with Google Sheets.
5) pdf2image:
   Converts PDF files into images so that we can extract text from them.
6) Pillow:
   A library used to work with images in Python.
7) Matplotlib:
   Used to display images on the screen.

Workflow :-

- Install Tools and Libraries:

  We install special tools like Tesseract OCR and Google Sheets API so the computer can read words and save them.

- Connect to Google Sheets:

  We connect the computer to Google Sheets so it can save the words it reads.

- Check the File Type:

  The computer checks if the invoice is a PDF or an image (like JPG or PNG).

- Convert PDF to Images:

  If the invoice is a PDF, the code turns each page into an image.

- Display the Image:

  The computer shows the image on the screen so we can see what it's reading.

- Extract Text with OCR:

  The computer uses Tesseract OCR to read the words from the image or PDF.

- Organize the Text:

  The computer splits the text into lines and words so it's easier to read.

- Save Text to Google Sheets:

  The computer saves the extracted words into Google Sheets in an organized way.

- Error Handler's:

  If something goes wrong, the computer returns us the error.

Code :-

```
!apt-get install -y tesseract-ocr poppler-utils -q
!pip install pytesseract -q
!pip install gspread -q
!pip install oauth2client -q
!pip install pillow -q
!pip install pdf2image -q
!pip install pytesseract -q
!pip install matplotlib -q

# Import Libraries
import gspread
```

```
from oauth2client.service_account import ServiceAccountCredentials
from pdf2image import convert_from_path
from PIL import Image
import pytesseract
import os
import matplotlib.pyplot as plt

# Function to Connect to Google Sheets
def connect_to_google_sheet(sheet_name):
    # Scope of the API
    scope = ["https://spreadsheets.google.com/feeds",
"https://www.googleapis.com/auth/drive"]

    # Authenticate using the credentials JSON file
    credentials = ServiceAccountCredentials.from_json_keyfile_name(
        "/content/drive/MyDrive/Colab Notebooks/task1.json", scope
    )
    client = gspread.authorize(credentials)

    # Open the Google Sheet
    sheet = client.open(sheet_name).sheet1
    return sheet


# Display Image on Console
def display_image(image):
    plt.figure(figsize=(8, 10))
    plt.imshow(image, cmap='gray')
    plt.axis('off')
    plt.show()

# Function to Check File Type
def get_file_type(file_path):
    file_extension = os.path.splitext(file_path)[1].lower()  # Get the
file extension
    if file_extension in ['.jpg', '.jpeg', '.png']:
        r   eturn 'image'
    elif file_extension == '.pdf':
        return 'pdf'
    else:
        raise ValueError("Unsupported file type. Please provide an image
or PDF.")

# Function to Convert PDF to Images
def convert_pdf_to_images(pdf_path):
    images = convert_from_path(pdf_path, dpi=300, grayscale=True,)
    return images

# Function to Extract Text with Tesseract
def extract_text_from_images(images):
    all_text = []
    for image in images:
        config = '--oem 3 --psm 6'  # Adjust configuration for better OCR
results
```

```
        text = pytesseract.image_to_string(image, config=config)
        all_text.append(text)
    return "\n".join(all_text)  # Combine all pages' text

# Write Text to Google Sheets
def write_text_to_sheet(sheet, text):

    lines = text.splitlines()  # Split the text into lines

    data = [line.split() for line in lines]  # Split each line into words
for separate columns

    # Clear the sheet first
    sheet.clear()

    # Update the sheet using batch_update
    sheet.batch_update([{'range': 'A1', 'values': data}])


# Function to Process File
def process_file(file_path, sheet_name):
    try:
        # Connect to Google Sheet
        sheet = connect_to_google_sheet(sheet_name)

        # Check if the file is a PDF or Image
        file_type = get_file_type(file_path)

        if file_type == 'image':
            image = Image.open(file_path)
            display_image(image)
            text = extract_text_from_images(image)
        elif file_type == 'pdf':
            # Convert PDF to Images
            pages = convert_pdf_to_images(file_path)
            for i, page in enumerate(pages):
                print(f"Displaying image from page {i+1}")
                display_image(page)
            text = extract_text_from_images(pages)

        # Write the extracted text to Google Sheets
        write_text_to_sheet(sheet, text)

        print("Text extracted and saved to Google Sheet successfully!")
    except Exception as e:
        print(f"Error: {e}")


# Call the Function
file_path = '/content/drive/MyDrive/Task3/invoices/printable_invoice.pdf'
sheet_name = 'Task7'
process_file(file_path, sheet_name)
```

Results :-

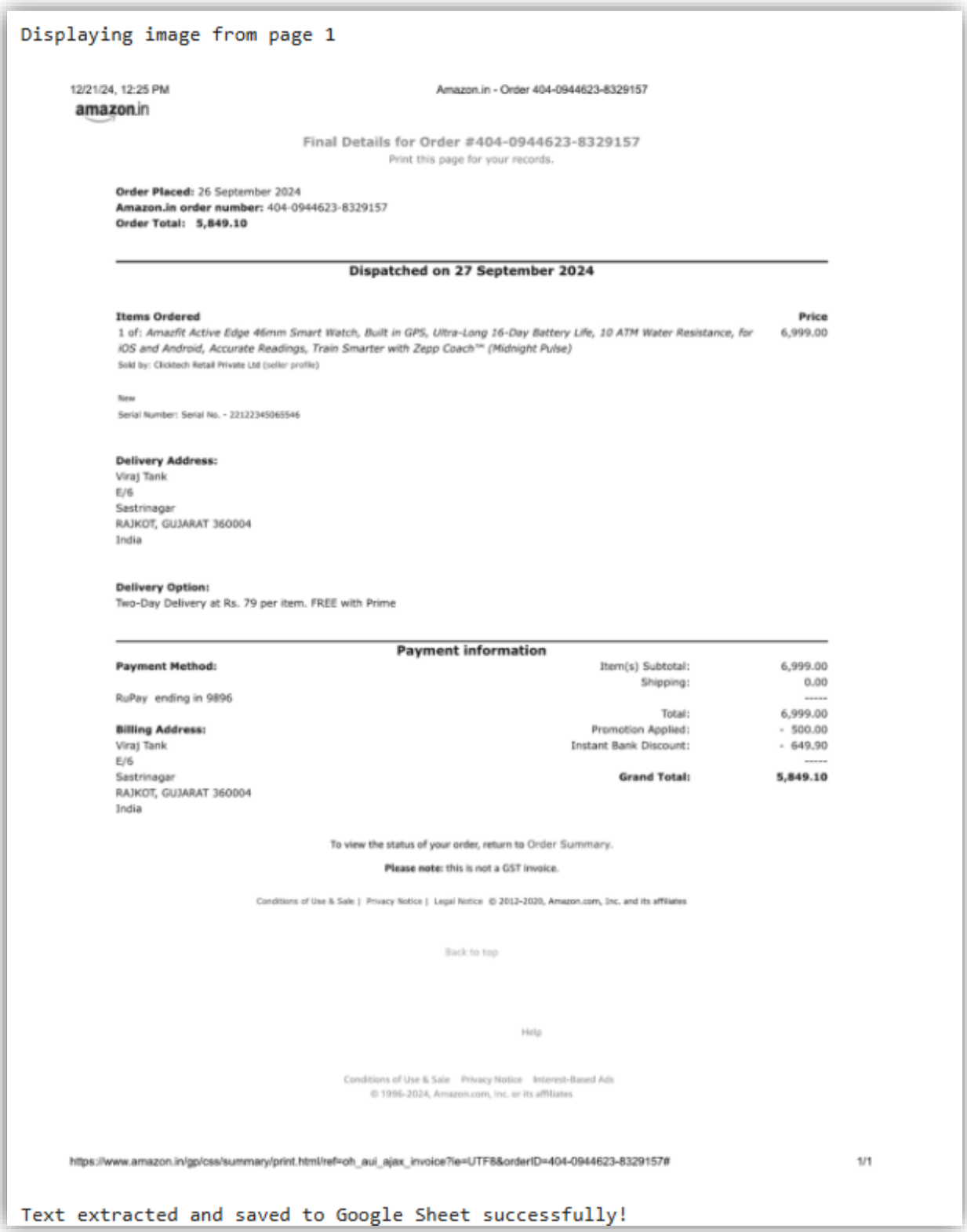
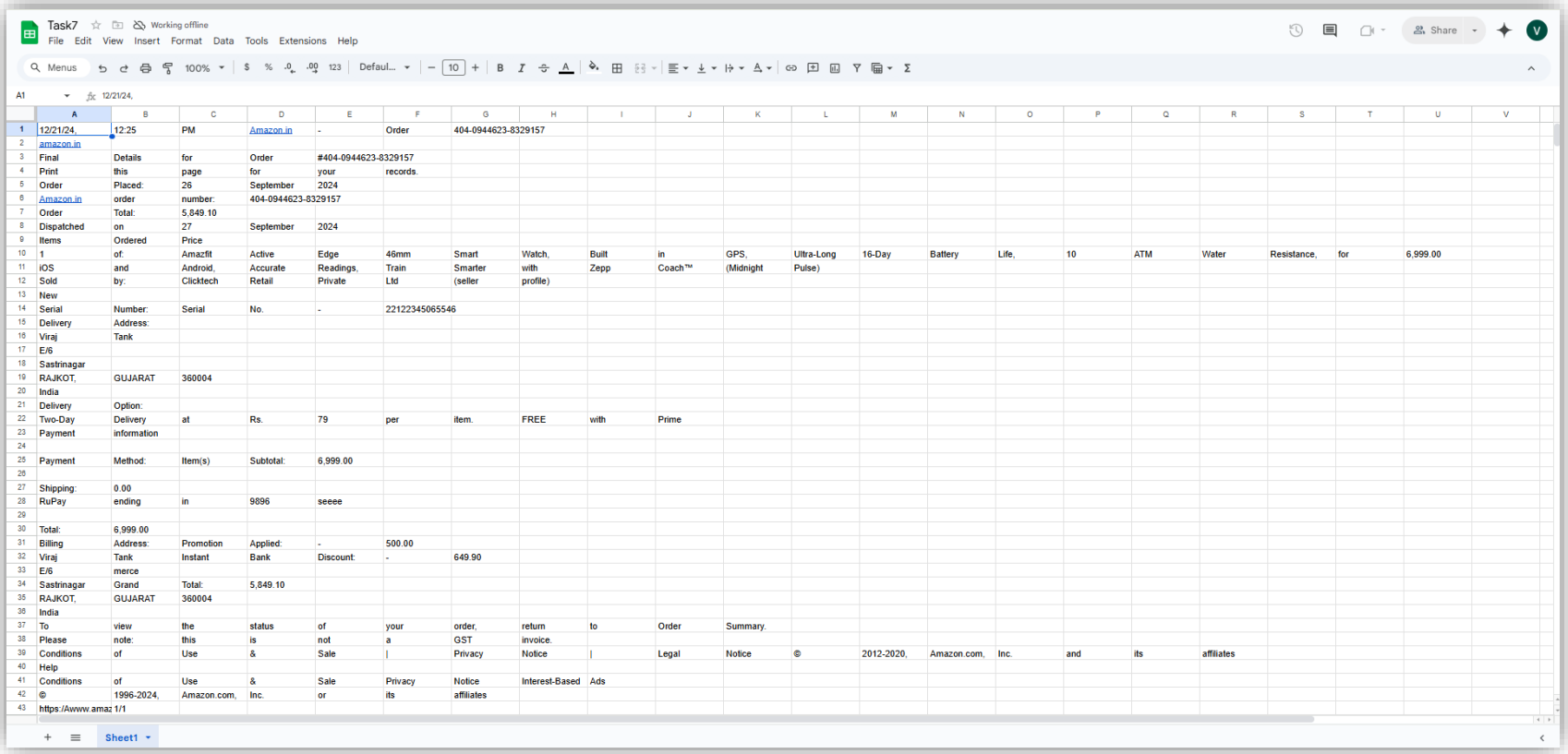


Fig 7.1 Input image displayed on Output Cell



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	12/21/24,	12:25	PM	Amazon.in	-	Order	404-0944623-8329157															
2	amazon.in																					
3	Final	Details	for	Order	#404-0944623-8329157																	
4	Print	this	page	for	your	records.																
5	Order	Placed:	26	September	2024																	
6	Amazon.in	order	number:	404-0944623-8329157																		
7	Order	Total:	5,849.10																			
8	Dispatched	on	27	September	2024																	
9	Items	Ordered	Price																			
10	1	of:	Amazfit	Active	Edge	46mm	Smart	Watch,	Built	in	GPS,	Ultra-Long	16-Day	Battery	Life,	10	ATM	Water	Resistance,	for	6,999.00	
11	iOS	and	Android,	Accurate	Readings,	Train	Smarter	with	Zepp	Coach™	(Midnight	Pulse)										
12	Sold	by:	Clicktech	Retail	Private	Ltd	(seller	profile)														
13	New																					
14	Serial	Number:	Serial	No.	-	22122345065546																
15	Delivery	Address:																				
16	Viraj	Tank																				
17	E/6																					
18	Sastrinagar																					
19	RAJKOT,	GUJARAT	360004																			
20	India																					
21	Delivery	Option:																				
22	Two-Day	Delivery	at	Rs.	79	per	item.	FREE	with	Prime												
23	Payment	information																				
24																						
25	Payment	Method:	Item(s)	Subtotal:	6,999.00																	
26																						
27	Shipping:	0.00																				
28	RuPay	ending	in	9896	seeee																	
29																						
30	Total:	6,999.00																				
31	Billing	Address:	Promotion	Applied:	-	500.00																
32	Viraj	Tank	Instant	Bank	Discount:	-	649.90															
33	E/6	merce																				
34	Sastrinagar	Grand	Total:	5,849.10																		
35	RAJKOT,	GUJARAT	360004																			
36	India																					
37	To	view	the	status	of	your	order,	return	to	Order	Summary.											
38	Please	note:	this	is	not	a	GST	invoice.														
39	Conditions	of	Use	&	Sale	|	Privacy	Notice	|	Legal	Notice	©	2012-2020,	Amazon.com,	Inc.	and	its	affiliates				
40	Help																					
41	Conditions	of	Use	&	Sale	Privacy	Notice	Interest-Based	Ads													
42	©	1996-2024,	Amazon.com,	Inc.	or	its	affiliates															
43	https://www.amaz	1/1																				

Fig 7.2 Extracted text on Google Sheets

Observation :-

- If the input file is an image, there is no problem in viewing the image on the console.
- However, when the input file is a PDF, converting it to an image sometimes makes the file appear blurred.
- Despite this, the text extracted from both the image and PDF remains the same and is pretty accurate compared to the original text.
- Viewing the results on Google Sheets proved to be a good step, as it makes the text easily visible and searchable.

Conclusion :-

In conclusion, the system works well for extracting text from both images and PDFs. Even though PDFs sometimes look blurry when converted to images, the text we get is still accurate. Saving the text into Google Sheets is a good idea because it makes everything easy to read and search.

However, there are a few issues. For example, PDFs can get blurry, and we still need to check the text manually to make sure there are no mistakes.

In the future, we plan to organize the extracted data better, like separating dates, amounts, and invoice numbers, so it can be used for more tasks. This will make the system even more helpful and efficient.

Task 8 :- Invoice Data Extraction and Organising Using OCR and Regex

Introduction :-

In this task, we'll use tools like Tesseract and PaddleOCR to read text from invoices, whether they're images or PDFs. Once the text is extracted, we'll use regex (regular expressions) to pull out important details like invoice numbers, dates, and amounts. After that, we'll organize the data neatly and save it as a JSON file, making it easy to access and use later. This helps make invoice processing quicker and more efficient.

Definition's :-

1) **Regex** :- Regex (Regular Expressions) is a powerful tool used to find specific patterns in text. It works like a search query but is much more flexible and precise. For example, if you want to extract an invoice number or an address from a block of text, regex can help you find exactly what you're looking for by defining a pattern. It's especially useful for pulling out structured information (like dates, amounts, or IDs) from unstructured text. To get a proper idea how regex works see below example's.
    - **r"Invoice Number\s*:\s*(\S+)"**

      This pattern looks for the phrase "Invoice Number", followed by a colon (😊 and optional spaces (\s*).

      (\S+): This part captures the actual invoice number, which is a sequence of non-whitespace characters (like INV-1234).

      If the text is "Invoice Number: INV-1234", the regex will extract INV-1234 as the invoice number as sequence of non-whitespace characters (like INV-1234).
    - **r"Billing Address\s*:\s*([\s\S]*?)\n\n"**

      This pattern looks for the phrase "Billing Address", followed by a colon (😊 and optional spaces (\s*).

([\s\S]*?): This part captures everything (including spaces and newlines) after the colon until it finds two newlines (\n\n), which usually marks the end of the address.

2) **JSON** :- JSON (JavaScript Object Notation) is a way to store information in a neat and organized format. Think of it like a labelled box where you can put things. We use JSON because it's easy to read and write, and it organizes data into clear key-value pairs. Unlike a plain text file, JSON tells you exactly what each piece of information means. It's also lightweight and works with almost every programming language, making it perfect for sharing data between computers or over the internet. For example, if you save an invoice number and date in a plain text file, it might look messy, but in JSON, it's clear and organized. This makes JSON a better choice for storing and sharing data.

Workflow :-

- Install Required Tools:
  Install necessary tools like Tesseract OCR, PaddleOCR, and Python libraries to process invoices.

- Load the Invoice:
  Load the invoice file, which can be an image (JPG, PNG).

- Extract Text Using OCR:
  Use Tesseract OCR and PaddleOCR to extract text from the invoice image. This step converts the visual content of the invoice into editable and usable text for further processing.

- Extract Specific Data Using Regex:
  Use regex patterns to find and extract specific details like invoice numbers, dates, addresses, and amounts from the cleaned text. Regex helps us locate these details by searching for predefined patterns in the text. This step ensures that only the relevant information is pulled out and organized for further use.

- Organize the Extracted Data:
  Structure the extracted data into key-value pairs using JSON. This makes the data easy to read and use.

- Save the Data to a JSON File:

  Save the organized data into a JSON file for easy access, sharing, or further processing.

- Display the Results:

  Display the extracted and organized data on the console or save it to a file for verification and use.

Code :-

```
!pip install pytesseract -q
!pip install pillow -q
!pip install paddleocr -q
!pip install paddlepaddle -q
!sudo apt install tesseract-ocr -q

# Import Libraries
import pytesseract
from PIL import Image
import re
import json
from paddleocr import PaddleOCR

# Function to extract text using Tesseract OCR
def extract_text_tesseract(image):
    return pytesseract.image_to_string(image)

# Function to extract text using PaddleOCR
def extract_text_paddleocr(image_path):
    ocr = PaddleOCR(use_angle_cls=True, lang='en')
    result = ocr.ocr(image_path)
    return "\n".join([line[1][0] for line in result[0]])

# Function to extract data using regex
def extract_data_using_regex(text, patterns):
    extracted_data = {}
    for field, pattern in patterns.items():
        match = re.search(pattern, text)
        extracted_data[field] = match.group(1).strip() if match else None
    return extracted_data

# Function to save the extracted data to JSON and display it on the
console
def save_to_json(data, file_path):
    with open(file_path, "w") as json_file:
        json.dump(data, json_file, indent=4)
    print("Data saved to json file")
    print(json.dumps(data, indent=4))

# Path to the image on Google Drive
```

```
image_path = “/content/drive/MyDrive/Task3/invoices/invoice_pages-to-jpg-0001.jpg”
output_file_path = “/content/drive/MyDrive/Colab Notebooks/task8.json”

# Load the image
image = Image.open(image_path)

# Extract text using Tesseract OCR
extracted_text_tesseract = extract_text_tesseract(image)

# Extract text using PaddleOCR
extracted_text_paddle = extract_text_paddleocr(image_path)

# Define regex patterns for fields
patterns = {
    “invoice_number”: r”Invoice Number\s*:\s*(\S+)”,
    “billing_address”: r”Billing Address\s*:\s*([\s\S]*?)\n\n”,
    “shipping_address”: r”Shipping Address\s*:\s*([\s\S]*?)\n\n”,
    “order_number”: r”Order Number\s*:\s*(\S+)”,
    “order_date”: r”Order Date\s*:\s*(\S+)”,
    “invoice_date”: r”Invoice Date\s*:\s*(\S+)”,
    “pan_no”: r”PAN No\s*:\s*(\S+)”,
    “gst_no”: r”GST Registration No\s*:\s*(\S+)”,
    “total_amount”: r”Total Amount\s*:\s*₹([\d,]+(?:\.\d{2})?)”,
}

# Organize data using regex from Tesseract OCR
extracted_data_tesseract =
extract_data_using_regex(extracted_text_tesseract, patterns)

# Organize data using regex from PaddleOCR
extracted_data_paddle = extract_data_using_regex(extracted_text_paddle,
patterns)


# Combine results from both OCRs
comparison_data = {
    “Tesseract OCR”: extracted_data_tesseract,
    “PaddleOCR”: extracted_data_paddle
}

# Save the extracted to .json file
save_to_json(comparison_data, output_file_path)
```

Results :-

Output on console :-

Fig 8.1 Data saved to Json file

Saved JSON File :-

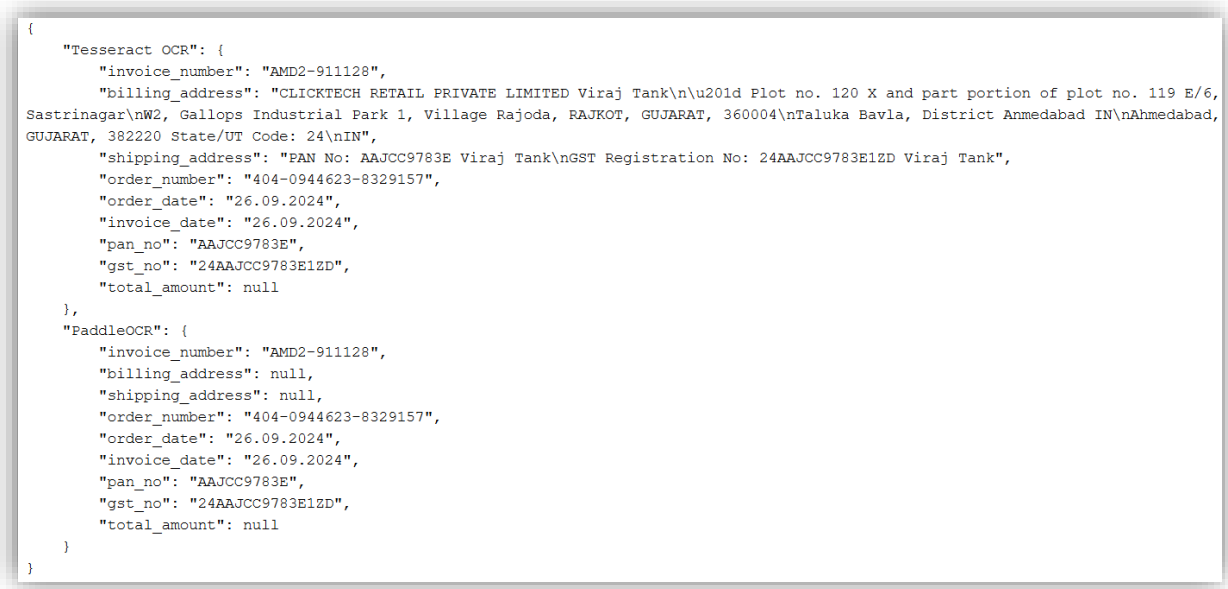


Fig 8.2 Json file

Observation :-

- For fields like billing address and shipping address, Tesseract OCR does not provide the desired output. The extracted text is also containing the text from its side column's which is not desired.
- In the case of billing address and shipping address, PaddleOCR returns null values, meaning because of two columns it fails to extract any data for these fields.

Conclusion :-

In conclusion, the system works well for extracting simple fields like invoice numbers and dates, but it struggles with multi-line content such as billing and shipping addresses. This is because the data in these fields is spread across multiple lines and sometimes arranged in different columns, making it harder for the OCR tools to extract accurately. While we can modify the code to handle these fields better, it would only work for invoices with a similar layout, like the Amazon invoice used in this case. This means the code would lose its flexibility to process different types of invoices. For now, the system is effective for structured data but requires improvements to handle more complex or varied invoice formats.