

A Weekly Report
on
INVOICE ASSISTANT
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.
(Feb 2025)

Task 9 :- Invoice Data Extraction Using Google Gemini

Introduction :-

In this task, we will use Google Gemini, a powerful AI model, to extract information from invoices. Unlike the previous task, where we could only work with one type of invoice (like Amazon invoices), Gemini can handle any kind of invoice—whether it's from a restaurant, an online store, or a utility bill. This makes it much more flexible and useful. Gemini can understand complex layouts, including multi-line text and tables, and extract details like supplier information, addresses, item lists, payment details, and taxes. The extracted data will be saved in a JSON file, making it easy to access and use for further tasks.

Definition's :-

- 1) **Google Gemini** :- Google Gemini is a smart AI tool that can read and pull out information from complicated documents like invoices. It's great at handling different layouts, such as multi-line text, tables, and various formats. In this task, we use the "gemini-1.5-flash" model, a faster and more accurate version of Gemini, to extract details from images quickly. The free version allows 15 requests per minute (RPM), 1 million tokens per minute (TPM), and 1,500 requests per day (RPD), making it a powerful and accessible tool for processing invoices efficiently.

Paste the API Key in the secret section of google collab.

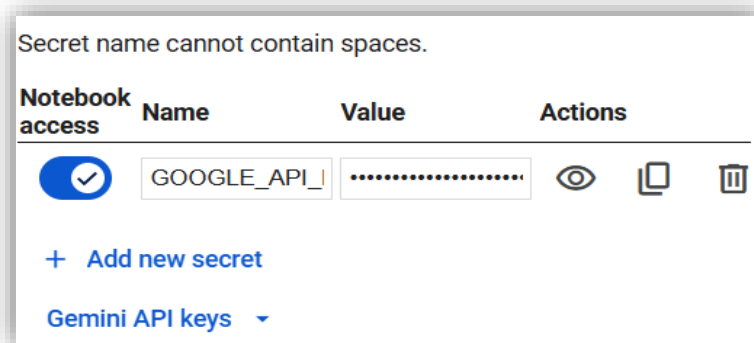


Fig 9.1 Secret Section of Collab

Configuration Process for Gemini API Key :-

- Visit to "<https://aistudio.google.com>".
- Or else you can do same from google cloud console.
- Click on Get API Key.

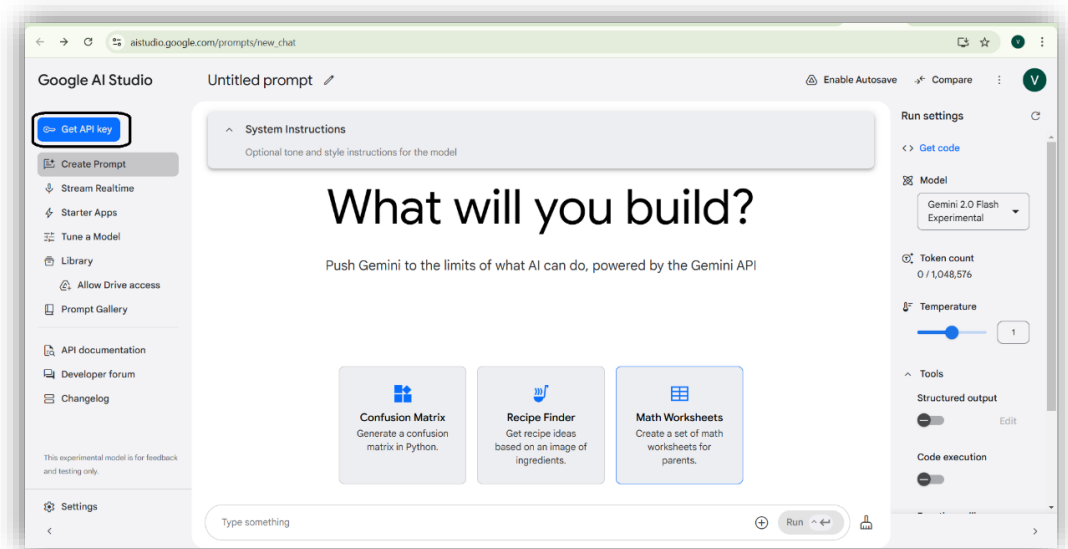


Fig 9.2 Get API Key

- Now Create API Key.



Fig 9.3 Create API Key

- Now copy the API Key for further use.



Fig 9.4 Copy API Key

Workflow :-

- **Install Required Libraries:**
Install necessary libraries like `google.generativeai`, `json`, and `matplotlib` to process invoices and visualize results.
- **Set Up Google Gemini:**
Configure the Google Gemini API using the provided API key to enable communication with the AI model.
- **Load the Invoice Image:**
Load the invoice image from the specified file path.
- **Prepare Image Data:**
Convert the invoice image into a format that can be processed by Google Gemini.
- **Extract Invoice Fields:**
Use the "gemini-1.5-flash" model to extract detailed information from the invoice, such as supplier details, billing and shipping addresses, item lists, payment details, and tax calculations.
- **Clean the Extracted Data:**
Remove unnecessary delimiters or formatting from the extracted text to prepare it for JSON conversion.

- **Convert Data to JSON:**
Convert the cleaned text into a structured JSON format for easy access and further use.
- **Save Data to JSON File:**
Save the extracted and organized data into a JSON file for easy access and sharing.
- **Display the Results:**
Print the extracted data on the console and display the input invoice image for verification.

Code :-

```
import json
import google.generativeai as genai
import os
import cv2
import matplotlib.pyplot as plt

# Set the API key directly
from google.colab import userdata
genai.configure(api_key=userdata.get('GOOGLE_API_KEY'))

# Function to Prepare Image For Google Gemini
def prepare_image_data(file_path):
    if not os.path.exists(file_path):
        raise FileNotFoundError(f"File not found: {file_path}")

    with open(file_path, "rb") as file:
        bytes_data = file.read()

    image_data = [
        {
            "mime_type": "image/jpeg",
            "data": bytes_data
        }
    ]
    return image_data

# Function for Initialization of Google Gemini
def extract_invoice_fields(image_data):
    input_prompt = """
        # Complete prompt is written in collab file please refer that.
        """

    model = genai.GenerativeModel('gemini-1.5-flash')

    response = model.generate_content([input_prompt, image_data[0]])
    return response.text
```

```

# Function to Save data to Json
def save_to_json(data, output_path):
    with open(output_path, "w") as json_file:
        json.dump(data, json_file, indent=4)

# Main Function
def main():

    # Input and Output File Paths
    #input_file_path =
    "/content/drive/MyDrive/Task3/invoices/invoice_pages-to-jpg-0001.jpg"
    input_file_path = "/content/drive/MyDrive/Task3/invoices/our-food-
bill-gives-an.jpg"
    output_file_path = "/content/drive/MyDrive/Colab Notebooks/task9.json"

    # Read and Print the Image
    img = cv2.imread(input_file_path)
    print("Input Invoice Image")
    plt.imshow(img)
    plt.axis('off')
    plt.show()

    try:
        # Prepare image data
        image_data = prepare_image_data(input_file_path)

        # Extract fields from invoice using Gemini
        response_text = extract_invoice_fields(image_data)

        # Cleaning the initial_response to remove delimiters and language
        specifier
        json_content = response_text.strip('```').replace('json\n', '',
1).strip()

        # Convert response text to JSON
        extracted_data = json.loads(json_content)

        # Save extracted data to a JSON file
        save_to_json(extracted_data, output_file_path)

        print("\nFields extracted successfully!")
        print(json.dumps(extracted_data, indent=4))
    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    main()

```

Results :-

- Test Image 1

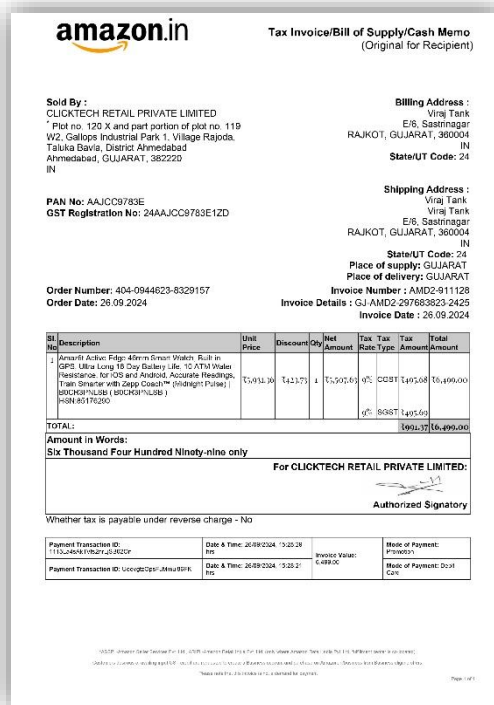


Fig 9.5 Input Image 1

Output of Image 1

Fields extracted successfully!

```
{
  "Supplier": {
    "Name": "CLICKTECH RETAIL PRIVATE LIMITED",
    "Address": "Plot no. 120 X and part portion of plot no. 119 W2, Gallops Industrial Park 1, Village Rajoda, Taluka Bavla, District Ahmedabad Ahmedabad, GUJARAT, 382220 IN",
    "PAN Number": "AAJCC9783E",
    "GST Number": "24AAJCC9783E1ZD",
    "Contact": null
  },
  "ReceiptDetails": {
    "Billing Address": "Viraj Tank E/6, Sastrinagar RAJKOT, GUJARAT, 360004 IN",
    "Shipping Address": "Viraj Tank E/6, Sastrinagar RAJKOT, GUJARAT, 360004 IN",
    "Invoice Number": "AMD2-911128",
    "Date": "2024-09-26",
    "Time": "15:28:29"
  },
  "Items": [
    {
      "ItemName": "Amazfit Active Edge 46mm Smart Watch, Built in GPS, Ultra-Long 16-Day Battery Life, 10 ATM Water Resistance, for iOS and Android, Accurate Readings, Train Smarter with Zepp Coach \u2012122 (Midnight Pulse) | B0CR3PNLSB (B0CR3PNLSB)",
      "Quantity": 1,
      "Unit Price": "5507.63",
      "TotalPrice": "6499.00"
    }
  ]
}
```

```

    ],
    "Payment Details": {
      "Payment Method": "Promotion/Debit Card",
      "Currency": "Indian Rupees",
      "Total Amount": "6499.00",
      "Taxes": "991.37",
      "Discounts": "423.73"
    },
    "Tax calculation": {
      "CGST": "495.68",
      "SGST": "495.69",
      "Total Tax": "991.37"
    }
  }
}

```

- Test Image 2



Fig 9.6 Input Image 2

Output of Image 2

Fields extracted successfully!

```

{
  "Supplier": {
    "Name": "HOTEL AMER PALACE RESTAURANT",
    "Address": "Hoshangabad Road, Ratanpur, Bhopal-462046, Madhya Pradesh",

```



```

    "GSTIN": "23AABFH6030L1ZN",
    "TIN": "23894007031",
    "Contact": null
  },
  "ReceiptDetails": {
    "Billing Address": null,
    "Shipping Address": null,
    "Invoice Number": "GRT1715",
    "Date": "2021-07-24",
    "Time": "15:14"
  },
  "Items": [
    {
      "ItemName": "VEG MANCHAW SO",
      "Quantity": 1,
      "Unit Price": "119.00",
      "TotalPrice": "119.00"
    },
    {
      "ItemName": "VEG SWEET CORN",
      "Quantity": 1,
      "Unit Price": "119.00",
      "TotalPrice": "119.00"
    },
    {
      "ItemName": "DAL TADKA",
      "Quantity": 1,
      "Unit Price": "215.00",
      "TotalPrice": "215.00"
    },
    {
      "ItemName": "JEERA RICE",
      "Quantity": 1,
      "Unit Price": "145.00",
      "TotalPrice": "145.00"
    },
    {
      "ItemName": "PLAIN PAPAD",
      "Quantity": 2,
      "Unit Price": "40.00",
      "TotalPrice": "80.00"
    },
    {
      "ItemName": "BAKED VEG WITH",
      "Quantity": 1,
      "Unit Price": "270.00",
      "TotalPrice": "270.00"
    },
    {
      "ItemName": "MUSHROOM MATTA",
      "Quantity": 1,
      "Unit Price": "265.00",
      "TotalPrice": "265.00"
    },
    {
      "ItemName": "BUTTER TANDOOR",
      "Quantity": 1,
      "Unit Price": "27.00",
      "TotalPrice": "27.00"
    },
    {
      "ItemName": "MISSI ROTI",
      "Quantity": 1,
      "Unit Price": "30.00",
      "TotalPrice": "30.00"
    }
  ],

```

```

    {
      "ItemName": "GARLIC NAAN",
      "Quantity": 1,
      "Unit Price": "45.00",
      "TotalPrice": "45.00"
    }
  ],
  "Payment Details": {
    "Payment Method": null,
    "Currency": "Indian Rupee",
    "Total Amount": "1381.00",
    "Taxes": "65.76",
    "Discounts": null
  },
  "Tax Calculation": {
    "CGST": "32.88",
    "SGST": "32.88",
    "Total Tax": "65.76"
  },
  "Subtotal": 1315.24
}

```

Observation :-

- Now, we are getting all the details properly, including the billing and shipping addresses.
- Some fields that are not present in the invoice are marked as null.

Conclusion :-

In this task, we successfully used Google Gemini to extract detailed information from invoices. Unlike the previous task, which was limited to specific invoice types (like Amazon invoices), this system can now handle any kind of invoice, whether it's from a restaurant, e-commerce platform, or utility bill. All fields, including billing and shipping addresses, are extracted correctly, and the data is saved in a JSON file for easy access and further use. This makes the system highly flexible and useful for various applications.

However, there is one drawback: the free version of Gemini allows only 15 requests per minute. This limitation needs to be considered when processing multiple invoices.

Task 10 :- Interactive Invoice Query System

Introduction :-

In this task, we will create a system where users can ask questions about an invoice and get instant answers. Using Google Gemini, a smart AI model, the system can read and understand the content of an invoice. Unlike earlier tasks, this system is interactive, meaning users can ask multiple questions—like "What is the invoice number?" or "What is the total amount?"—and get quick, conversational answers. This makes it a simple and user-friendly tool for understanding invoice details without any hassle.

Workflow :-

- **Install Required Libraries:**
Install necessary libraries like `google.generativeai`, `json`, and `matplotlib` to process invoices and visualize results.
- **Set Up Google Gemini:**
Configure the Google Gemini API using the provided API key to enable communication with the AI model.
- **Load the Invoice Image:**
Load the invoice image from the specified file path.
- **Prepare Image Data:**
Convert the invoice image into a format that can be processed by Google Gemini.
- **Ask User Questions:**
Prompt the user to ask questions about the invoice, such as the invoice number, billing address, or total amount.
- **Get Gemini's Response:**
Use the "gemini-1.5-flash" model to generate answers to the user's questions based on the invoice content.
- **Display the Response:**

Show the response to the user in a conversational format.

- Continue or Exit:

Ask the user if they want to ask another question. If yes, repeat the process; if no, end the session.

Code :-

```
import os
from PIL import Image
import google.generativeai as genai

# Set the API key directly
from google.colab import userdata
genai.configure(api_key=userdata.get('GOOGLE_API_KEY'))

# Function to Load the Gemini model and get a response
def get_gemini_response(input_prompt, image, question):
    model = genai.GenerativeModel('gemini-1.5-flash')
    response = model.generate_content([input_prompt, image[0], question])
    return response.text

# Function to Load and process the image
def input_image_setup(image_path):
    if not os.path.exists(image_path):
        raise FileNotFoundError(f"The file does not exist.")

    # Open the image and convert to bytes
    with open(image_path, "rb") as file:
        bytes_data = file.read()

    image_parts = [
        {
            "mime_type": "image/jpeg",
            "data": bytes_data
        }
    ]
    return image_parts

# Main Function
def main():
    image_path = "/content/drive/MyDrive/Task3/invoices/invoice_pages-to-jpg-0001.jpg"

    try:
        image_data = input_image_setup(image_path)
    except FileNotFoundError as e:
        print(e)
        return
```

```

input_prompt = """
You are an expert in understanding invoices.
You will receive input images as invoices &
You are given with the data and user query.
You just need to answer based on the information provided.
Answer in a conversational manner, as if talking to a human.
Any questions outside the information in the invoice will be
ignored and not answered.
Thank you!
"""

while True:
    # Ask the user for a question
    print("\nAsk a question about the invoice:")
    question = input()

    # Get the response from the Gemini model
    response = get_gemini_response(input_prompt, image_data, question)

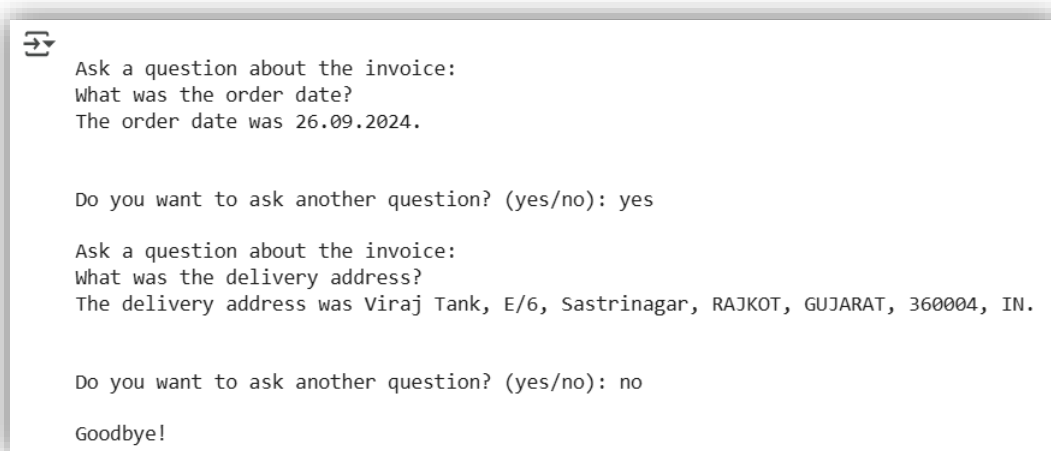
    # Display the response
    print(response)

    # Ask if the user wants to ask another question
    continue_prompt = input("\nDo you want to ask another question?
(yes/no): ").strip().lower()
    if continue_prompt not in ("yes", "y"):
        print("\nGoodbye!")
        break

# Run the program
if __name__ == "__main__":
    main()

```

Results :-



```

Ask a question about the invoice:
What was the order date?
The order date was 26.09.2024.

Do you want to ask another question? (yes/no): yes

Ask a question about the invoice:
What was the delivery address?
The delivery address was Viraj Tank, E/6, Sastrinagar, RAJKOT, GUJARAT, 360004, IN.

Do you want to ask another question? (yes/no): no

Goodbye!

```

Fig 10.1 Interactive query and answer's

Observation :-

- We can ask multiple Questions about the invoice and the system answer's perfectly.
- The system works well for real-time queries, making it a practical tool for quickly extracting and understanding invoice details.
- It provides accurate responses to questions like order date, delivery address, and other invoice details.

Conclusion :-

In this task, we created an interactive system that allows users to ask questions about an invoice and get instant, accurate answers. Using Google Gemini, the system can understand and respond to queries like order date, delivery address, and total amount in a conversational manner. Unlike previous tasks, this system just focuses on a particular query asked by the user. So the system will only provide the data which user ask's and not all the data which might be a case for some user's.

Next, we will combine all the tasks to create a complete and proper project. This integrated system will automate the entire process—from extracting and organizing invoice data to allowing users to interactively query and understand the details—making it a powerful tool for businesses and individuals.