Final Report

1. Introduction

This project aims to automate the extraction, summarization, and analysis of text from PDF documents using various tools and techniques, including OCR (Optical Character Recognition) and natural language processing (NLP) models. The project involves extracting text from a PDF, generating summaries, flashcards, search queries, and fetching relevant images.

The motivation for this project was to streamline the process of information extraction from scanned or non-text-based PDFs, enabling the user to derive meaningful insights and educational content efficiently.

2. Initial Approach

The initial approach was to:

1. **Extract Text from PDF:** Use the pdf2image library to convert PDF pages into images and pytesseract for OCR to extract text from these images.
2. **Preprocess Images:** Convert the images to grayscale, reduce noise, and binarize them to improve OCR accuracy.
3. **Summarize Text:** Utilize OpenAI's GPT model to summarize extracted text.
4. **Generate Flashcards:** Create educational flashcards based on the extracted text.
5. **Create Search Queries for Images:** Formulate search queries for images related to the content.
6. **Fetch Relevant Images:** Scrape Google Images to retrieve URLs of relevant images.

3. Iterations and Challenges

Iteration 1: Basic Text Extraction

- **Objective:** The initial step focused on converting PDF pages to images using pdf2image and extracting text using pytesseract.
- **Challenges Faced:**

    - Low-quality OCR results due to noise, improper lighting, and non-standard fonts.
    - Difficulty handling PDFs with multiple columns or unusual layouts.

Iteration 2: Image Preprocessing for Improved OCR

- **Objective:** Enhance image quality to improve OCR accuracy.
- **Actions Taken:** Developed the preprocess_image function to:
    - Convert images to grayscale.
    - Apply median blurring to reduce noise.
    - Use Otsu's thresholding for binarization.
- **Outcome:** This significantly improved OCR results, reducing errors related to noise and lighting.

Iteration 3: Text Summarization Using GPT

- **Objective:** Summarize extracted text to provide concise overviews of the content.
- **Actions Taken:**
    - Utilized OpenAI's GPT-3.5-turbo model to summarize text.
    - Formulated a prompt to handle missing or incomplete words due to OCR limitations.
- **Challenges Faced:** Some summaries were less relevant due to the noisy input text.
- **Outcome:** Improved summaries were generated by refining the prompt and cleaning the extracted text.

Iteration 4: Flashcard Generation

- **Objective:** Create flashcards based on the content to facilitate learning and retention.
- **Actions Taken:**
    - Developed a function to interact with GPT to generate flashcards from the cleaned text.

    - Each flashcard included a question and answer format.
- **Outcome:** Effective flashcards were generated, though some context was missing in questions due to ambiguous text extraction.

Iteration 5: Generating Search Queries for Images

- **Objective:** Formulate search queries for images related to the text content.
- **Actions Taken:**
    - Used OpenAI's GPT to generate concise and specific search queries based on the text.
    - Focused on key concepts, objects, or themes mentioned in the text to avoid overly generic terms.
- **Outcome:** The search queries became more targeted, increasing the relevance of the fetched images.

Iteration 6: Fetching Images from Google

- **Objective:** Retrieve image URLs based on the search queries.
- **Actions Taken:**
    - Scraped Google Images using requests and BeautifulSoup libraries.
    - Extracted image URLs, limited to a maximum of 10 relevant images per query.
- **Challenges Faced:**
    - Managing HTTP requests and ensuring compliance with Google's scraping policies.
    - Filtering out irrelevant or placeholder images.
- **Outcome:** Successfully curated image URLs, with a few challenges in ensuring image relevance and quality.

4. Final Solution

The final solution integrates all the functions developed during the iterations into a single process that:

1. **Extracts Text:** Converts PDF pages to images and uses OCR to extract text.
2. **Preprocesses Text:** Cleans extracted text for better readability.
3. **Generates Summaries and Flashcards:** Uses GPT models to create summaries and flashcards.
4. **Creates Search Queries and Fetches Images:** Formulates search queries and scrapes Google Images to find relevant images.

5. Conclusion

This project illustrates a systematic approach to extracting and analyzing text from PDFs, integrating image preprocessing, OCR, NLP models, and web scraping techniques. The iterations helped refine each step, improving accuracy and relevance in text extraction, summarization, flashcard generation, and image fetching.

6. Future Improvements

- **Enhance OCR Accuracy:** Use advanced OCR models or integrate machine learning techniques to handle complex layouts better.
- **Improve Image Fetching:** Utilize APIs for fetching images more effectively and legally.
- **Expand NLP Capabilities:** Explore using other models or fine-tune existing models for better summarization and flashcard generation.

By continuously iterating and refining each stage, the project has achieved its objective of automating the extraction and analysis of text from PDF documents in a comprehensive and scalable manner.

Code snippet:

```
from pdf2image import convert_from_path
import cv2
import numpy as np
import pytesseract
import openai
import re
import requests
import json

from bs4 import BeautifulSoup

# Set your OpenAI API key
openai.api_key = "sk-proj-6PNhWH5iA1GZgTZc_P_12QMHxeXnEOclFdOsZzfkP4psJH3N1gYCHMRiOAKkUt6-
1g831sv3nTT3BlbkFJMo4ccxXxRa1Jp2_CSEqrGl7KLc3ncJtMJbpkTafG3iy5ZcX-nSJEI3TMEhEnIvMaXPPfMR_bQA"

# Function to preprocess images before OCR
def preprocess_image(image):
  grayscale = cv2.cvtColor(np.array(image), cv2.COLOR_BGR2GRAY)
  noise_reduced = cv2.medianBlur(grayscale, 3)
  binarized = cv2.threshold(noise_reduced, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]
  return binarized

# Function to extract text from a PDF using OCR
def extract_text_from_pdf(pdf_path):
  # Specify the path to Poppler binaries
  images = convert_from_path(pdf_path, poppler_path=r"C:\poppler-24.07.0\Library\bin")
  text = []
  for i, image in enumerate(images):
    preprocessed_image = preprocess_image(image)
    ocr_text = pytesseract.image_to_string(preprocessed_image)
    if ocr_text:
      text.append(ocr_text)
    else:
      text.append(f"No text extracted from page {i + 1}")
  return text

# Your other functions remain the same...
# Function to clean the extracted text
def clean_text(text):
  text = re.sub(r'\s+', ' ', text)  # Remove excessive whitespace
  text = re.sub(r'[^A-Za-z0-9.,;:?!()\'"\s]', '', text)  # Remove special characters
  return text.strip()

# Function to generate a summary using OpenAI
def generate_summary(page_text):
  response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[
      {"role": "system", "content": "You are a helpful assistant."},
      {"role": "user", "content": f"understand the text first and if the words are missing guess the relevent word and summarise the text:\n\n{page_text[:1000]}"}
    ]
  )
  summary = response['choices'][0]['message']['content']
  return summary

# Function to generate flashcards using OpenAI
def generate_flashcards(page_text):
  response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
```

```
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": f"Generate three flashcards (questions and answers) based on the text below. Each flashcard should be on a separate line, with the question on the first line and the answer on the second line:\n\n{page_text[:1000]}"}
        ]
    )
    flashcards = response['choices'][0]['message']['content']
    return flashcards

# Function to generate a search query using OpenAI
def generate_search_query(page_text):
    # Modify the prompt to provide more context and request a more specific query
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a helpful assistant specialized in generating search queries for images."},
            {"role": "user", "content": f"Based on the following text, create a concise and specific search query to find relevant images. Focus on key concepts, objects, or themes mentioned in the text, and include important keywords that would lead to relevant image results. Avoid overly generic terms:\n\n{page_text[:1000]}"}
        ]
    )
    search_query = response['choices'][0]['message']['content']
    return search_query

# Function to fetch image URLs from Google Images
def fetch_image_urls(query):
    query = query.replace(' ', '+')
    url = f"https://www.google.com/search?hl=en&tbm=isch&q={query}"
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.102 Safari/537.36"
    }

    try:
        response = requests.get(url, headers=headers)
        response.raise_for_status()
        soup = BeautifulSoup(response.text, 'html.parser')
        image_elements = soup.find_all('img')

        # Extract URLs, ensuring only relevant images (not icons or placeholders)
        image_urls = []
        for img in image_elements:
            if 'src' in img.attrs and img['src'].startswith('http'):
                image_urls.append(img['src'])
            if len(image_urls) >= 10:  # Limit to 10 images
                break

        return image_urls
    except requests.exceptions.RequestException as e:
        print(f"Error fetching images: {e}")
        return []

# Function to process the PDF and store the results
def process_and_store_results(pdf_path, output_summary_file, output_flashcards_file, output_queries_file):
    extracted_text = extract_text_from_pdf(pdf_path)
    print(f"Extracted text from PDF: {extracted_text}")

    # Prepare dictionaries to store results
    summaries = {}
    flashcards_dict = {}
    queries_and_images = {}

    for i, page_text in enumerate(extracted_text):
```

```
        print(f"\nProcessing Page {i + 1}...")

        page_id = f"Page {i + 1}"
        summaries[page_id] = {}
        flashcards_dict[page_id] = {}
        queries_and_images[page_id] = {}

        if "No text extracted" in page_text:
            print(page_text)
            summaries[page_id]["summary"] = page_text
            flashcards_dict[page_id]["flashcards"] = "No flashcards generated"
            queries_and_images[page_id]["query"] = "No query generated"
            queries_and_images[page_id]["images"] = "No images found"
            continue

        clean_page_text = clean_text(page_text)
        print(f"Cleaned Page Text: {clean_page_text}")

        # Generate summary
        summary = generate_summary(clean_page_text)
        print(f"Generated Summary: {summary}")
        summaries[page_id]["summary"] = summary

        # Generate flashcards
        flashcards = generate_flashcards(clean_page_text)
        print(f"Generated Flashcards: {flashcards}")
        flashcards_dict[page_id]["flashcards"] = flashcards

        # Generate search query
        search_query = generate_search_query(clean_page_text)
        print(f"Generated Search Query: {search_query}")
        queries_and_images[page_id]["query"] = search_query

        # Fetch image URLs
        image_urls = fetch_image_urls(search_query)
        print(f"Curated Image URLs: {image_urls}")
        queries_and_images[page_id]["images"] = image_urls

    # Write summaries to JSON file
    with open(output_summary_file, 'w', encoding='utf-8') as file:
        json.dump(summaries, file, ensure_ascii=False, indent=4)

    # Write flashcards to JSON file
    with open(output_flashcards_file, 'w', encoding='utf-8') as file:
        json.dump(flashcards_dict, file, ensure_ascii=False, indent=4)

    # Write search queries and images to JSON file
    with open(output_queries_file, 'w', encoding='utf-8') as file:
        json.dump(queries_and_images, file, ensure_ascii=False, indent=4)

# Path to your PDF file and output JSON files
pdf_path = r"C:\Users\vedap\Downloads\CVS.pdf"
output_summary_file = r"C:\Users\vedap\OneDrive\Desktop\data project\output_summaries2.json"
output_flashcards_file = r"C:\Users\vedap\OneDrive\Desktop\data project\output_flashcards2.json"
output_queries_file = r"C:\Users\vedap\OneDrive\Desktop\data project\output_queries2.json"

# Process the PDF and store results in JSON files
process_and_store_results(pdf_path, output_summary_file, output_flashcards_file, output_queries_file)
```

C: > Users > vedap > Downloads > {} output_summaries.json > ...


```
{
    "Page 1": {
        "summary": "No text extracted from page 1"
    },
    "Page 2": {
        "summary": "Based on the available text and using context clues to guess some missing words, the text seems to summarize the electrical co
    },
    "Page 3": {
        "summary": "It seems like the text you provided is a mix of jumbled words and technical terms related to cardiac physiology and pressure-
    },
    "Page 4": {
        "summary": "Autoregulation refers to how blood flow to an organ remains constant over a wide range of perfusion pressures. Different organ
    },
    "Page 5": {
        "summary": "It seems that the text you provided is related to the cardiovascular system and describes the movement of fluids through capil
    },
    "Page 6": {
        "summary": "The text discusses various aspects related to arterial blood pressure. It defines arterial blood pressure as the lateral press
    },
    "Page 7": {
        "summary": "It seems like the text is a mix of words and abbreviations related to various topics including business, finance, and possibly
    },
    "Page 8": {
        "summary": "No text extracted from page 8"
    },
    "Page 9": {
        "summary": "It appears that the text is a mix of random words and phrases without clear meaning or context. It seems to be a jumble of wor
    },
    "Page 10": {
        "summary": "It seems that some words are missing from the text. Based on the remaining words and the context, it appears to be discussing
    },
    "Page 11": {
        "summary": "It appears that some words are missing from the text, making it difficult to provide an accurate summary. However, based on th
    },
```



```
{
    "Page 1": {
        "flashcards": "No flashcards generated"
    },
    "Page 2": {
        "flashcards": "What is the function of the SA node in the heart?\nThe SA node is the dominant pacemaker that generates impulses at the high
    },
    "Page 3": {
        "flashcards": "What is the purpose of pressure volume loops in cardiac assessment?\nPressure volume loops are used to assess the cardiac fu
    },
    "Page 4": {
        "flashcards": "What is autoregulation and how does it work in the body?\nBlood flow to an organ remains constant over a wide range of perfu
    },
    "Page 5": {
        "flashcards": "What determines fluid movement through capillaries? \nCapillary hydrostatic pressure, interstitial hydrostatic pressure, pla
    },
    "Page 6": {
        "flashcards": "What is arterial blood pressure defined as?\nArterial Blood Pressure GF : defined as the lateral pressure exerted by the col
    },
    "Page 7": {
        "flashcards": "Q) What are some possible side effects of medications mentioned in the text?\nA) Ineffective, Profound, or Tonic.\nQ) What d
    },
    "Page 8": {
        "flashcards": "No flashcards generated"
    },
    "Page 9": {
        "flashcards": "Q: What is the character of the text?\nA: CHARACTER.\n\nQ: What is the radiation af?\nA: RADIATION.\n\nQ: What is Ergosinn?\
    },
    "Page 10": {
        "flashcards": "What condition is being discussed in the text?\nAngina with associated Vasospasm.\n\nWhat phenomenon is described as Reyraud
    },
    "Page 11": {
        "flashcards": "What are the symptoms of the condition mentioned in the text?\nSostect dy au sens  do  uP 9 oe ie \n\nWhat is the X-ray find
```

```
{
    "Page 1": {
        "query": "No query generated",
        "images": "No images found"
    },
    "Page 2": {
        "query": "Search query: Anatomy of cardiac conduction system including SA node, AV node, bundle of His, bundle branches, and Purkinje fibers
        "images": []
    },
    "Page 3": {
        "query": "\"Cardiac pressure volume loops and valvular function in human heart\"",
        "images": []
    },
    "Page 4": {
        "query": "Search query: Autoregulation blood flow organ constant perfusion pressures vasoconstriction hypoxia vasodilators lung kidney skele
        "images": [
            "https://www.google.com/logos/doodles/2024/paris-games-archery-day-2-6753651837110537-s.png",
            "https://www.gstatic.com/ui/v1/menu/light_thumbnail2.png",
            "https://www.gstatic.com/ui/v1/menu/dark_thumbnail2.png",
            "https://www.gstatic.com/ui/v1/menu/device_default_thumbnail2.png"
        ]
    },
    "Page 5": {
        "query": "heart capillaries exchange in capillaries fluid movement through capillaries images regulatory forces in capillaries interstitial
        "images": []
    },
    "Page 6": {
        "query": "arterial blood pressure measurement arterial blood pressure difference between systolic and diastolic blood pressures mean arteria
        "images": []
    },
    "Page 7": {
        "query": "Search query: \"Abstract regal symbols and motifs in art and culture\"",
        "images": []
```