

Health Summary Generator – Detailed Project Plan

Project Selected:

Project 1: Health Summary Generator

1. Approach

The primary objective is to convert an unstructured, scanned PDF document containing veterinary medical records into a structured, summarized PDF health summary. This involves multiple stages, each focused on accurately extracting, categorizing, and formatting the necessary information.

Phases

1. Ingestion and OCR:

- **Objective:** Convert the unselectable PDF content into machine-readable text.
- **Process:** Use OCR on the scanned document to extract all visible text.

2. Data Parsing and Information Extraction:

- **Objective:** Identify and capture relevant fields, including client, patient, clinic, and medical records.
- **Process:** Utilize regular expressions, NLP, and custom parsing rules to segment extracted text into categories.

3. Data Structuring and Summarization:

- **Objective:** Organize extracted data into a structured format and prepare it for inclusion in the final PDF document.
- **Process:** Map fields to the required output template to ensure all sections are filled accurately.

4. PDF Generation:

- **Objective:** Create a polished, structured summary PDF in the required format.
- **Process:** Use a PDF generation library to lay out and format the extracted data into a clean document.

2. Technology Choices

• Programming Language: Python

- **Reason:** Python provides robust libraries for OCR, PDF manipulation, and text processing, and it's well-suited for quick iteration and scalability.

• OCR Tool: Tesseract OCR

- **Reason:** Tesseract is highly effective for text extraction from images, supports multiple languages, and integrates seamlessly with Python. It's open-source and efficient for high-quality OCR, essential for scanned documents.

• PDF Handling: PDFPlumber

Health Summary Generator – Detailed Project Plan

- **Reason:** PDFPlumber is effective for extracting text from PDF documents, especially useful if parts of the document are selectable, reducing OCR processing requirements.
 - **NLP and Text Parsing: spaCy and Regular Expressions**
 - **Reason:** spaCy is efficient for named entity recognition (NER), making it possible to extract client and patient details. Regular expressions provide the precision needed to parse specific sections and patterns in unstructured text, such as dates or marker names in medical records.
 - **Data Structuring: Pandas**
 - **Reason:** Pandas offers excellent data handling capabilities, which help organize and manipulate extracted information before structuring it in the output document.
 - **PDF Generation: ReportLab**
 - **Reason:** ReportLab provides powerful PDF generation tools, including the ability to format text, create tables, and manage complex layouts, which are crucial for creating a clean, structured summary PDF.
-

3. Solution Design

Module 1: Document Ingestion and OCR

- **OCR Conversion:**
 - Load each page of the PDF, apply OCR on non-selectable text, and store it as plain text.
 - If PDFPlumber identifies selectable text, skip OCR to save processing time.

```
import pdfplumber
import pytesseract
from PIL import Image

def extract_text_from_pdf(pdf_path):
    text_data = []
    with pdfplumber.open(pdf_path) as pdf:
        for page in pdf.pages:
            text = page.extract_text()
            if not text: # If text is not selectable, use OCR
                image = page.to_image(resolution=300).convert('RGB')
                text = pytesseract.image_to_string(image)
            text_data.append(text)
    return " ".join(text_data)
```

Health Summary Generator – Detailed Project Plan

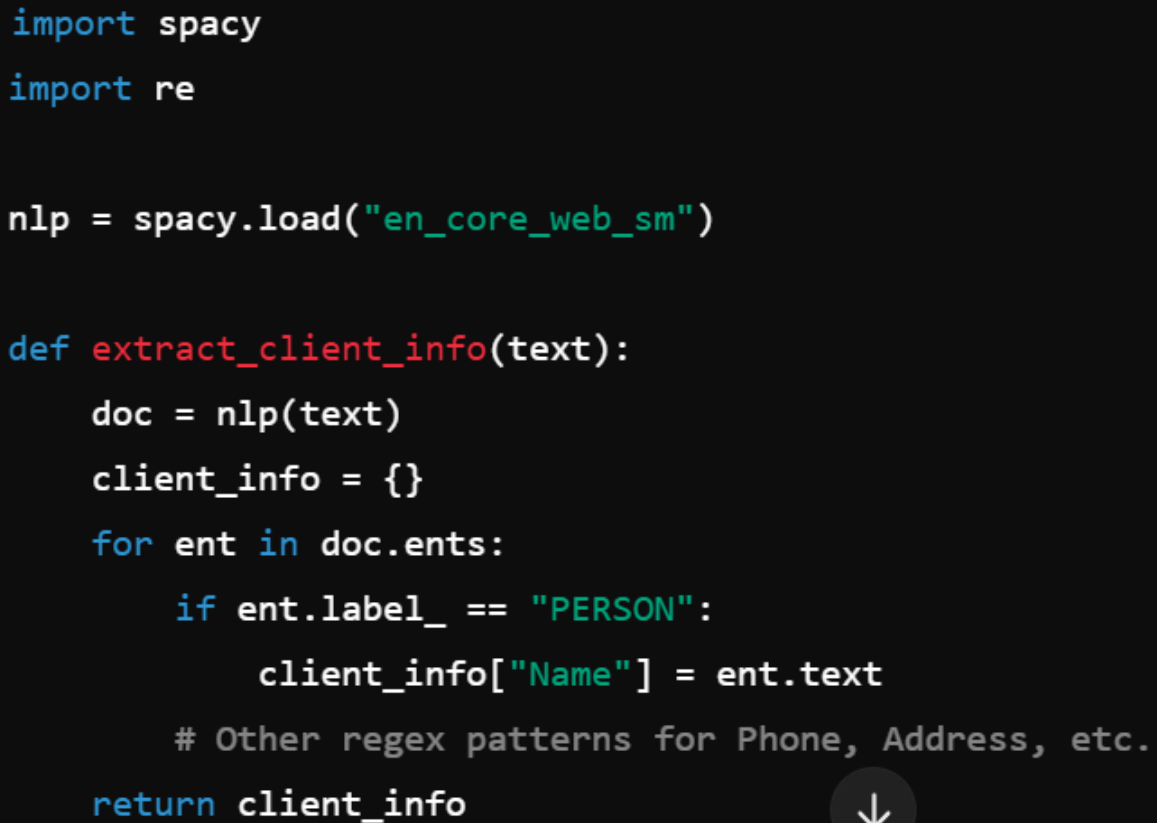
Module 2: Information Extraction and Categorization

- **Client and Patient Information Extraction:**
 - Use spaCy to detect entities like names and dates.
 - Apply regular expressions to capture structured fields such as "Name," "Breed," "Phone."

```
import spacy
import re

nlp = spacy.load("en_core_web_sm")

def extract_client_info(text):
    doc = nlp(text)
    client_info = {}
    for ent in doc.ents:
        if ent.label_ == "PERSON":
            client_info["Name"] = ent.text
    # Other regex patterns for Phone, Address, etc.
    return client_info
```



- **Medical Record Extraction:**
 - Divide text into sections based on known keywords ("Vaccinations," "Medications").
 - Parse specific data points within each section, such as medication strength and lab test dates.

Module 3: Structuring Data for Output

- **Data Mapping:**
 - Organize extracted fields into a dictionary to match the output template structure.
 - Use Pandas for tabular sections like "Lab Results" to ensure data consistency.

Health Summary Generator – Detailed Project Plan

```
import pandas as pd

def organize_medical_records(vaccination_data, medication_data):
    summary = {
        "Vaccinations": pd.DataFrame(vaccination_data),
        "Medications": pd.DataFrame(medication_data),
        # Add other sections similarly
    }
    return summary
```

Module 4: PDF Generation

- **Generate the Final Structured PDF:**
 - Layout each section using ReportLab, with specific page placements for each field.

```
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

def generate_health_summary_pdf(data):
    pdf = canvas.Canvas("Health_Summary.pdf", pagesize=letter)
    pdf.drawString(100, 750, "Health Summary Document")
    # Additional drawing functions for each section
    pdf.showPage()
    pdf.save()
```

4. Challenges and Solutions

- **OCR Accuracy with Scanned Documents:**
 - **Challenge:** OCR might not capture text accurately due to noise in scanned images.
 - **Solution:** Preprocess images before OCR (e.g., de-skewing, adjusting brightness/contrast) and use language-specific OCR tuning if necessary.
- **Unstructured Text Parsing:**
 - **Challenge:** Sections may not follow a fixed format across different documents.
 - **Solution:** Design flexible regex patterns to detect sections and fallback methods if expected patterns are missing.
- **Data Structuring Consistency:**

Health Summary Generator – Detailed Project Plan

- **Challenge:** Ensuring each section consistently maps to the template structure.
 - **Solution:** Implement data validation to check for missing or incorrect entries in each structured section.
 - **Formatting in PDF Generation:**
 - **Challenge:** PDF layout management for long text sections and tabular data.
 - **Solution:** Define a reusable layout style for each section and dynamically adjust positions for longer sections.
-

5. Assumptions

- **Consistent Section Headings:** It is assumed that all documents will have recognizable headings like “Client Information,” “Vaccinations,” etc.
 - **Limited Variance in Field Names:** Fields like “Phone,” “Breed,” etc., will appear with consistent naming conventions.
 - **Moderate Document Length:** Documents are assumed to be under 100 pages; larger documents may require optimization.
-

Additional Information

Success Metrics:

- **Accuracy:** Validate the extracted data by comparing summaries to original documents and measuring field accuracy.
- **Processing Time:** Aim to process each document within 1-2 minutes.

Validation Plan:

- Use sample documents to test and validate the extraction and PDF generation, adjusting parsing logic as needed based on results.

Enhancement Opportunities:

- Develop an interactive user interface for previewing extracted data before final PDF generation, allowing for any manual adjustments if needed.
- Can you some pretrained LLM and GenAI models to training the model to increase model accuracy by providing the large amount of data for training.
- Can integrate with GeminiAI or OpenAI to add some prescription or initial info to the doctor which has been given by the AI models from the output we have generated.

Past Experience and Relevant Knowledge:

- While I was doing my project “[Mental-Health-Chatbot](#)” I have to use the NLP tools and SVM to understand what the customer is saying and map particular token to generate response.