Curacel Take-Home Task: Intelligent Claims QA Service

Overview

At Curacel, we regularly process scanned or photographed insurance claim documents to verify treatments, medications, and other provided services. These documents are often unstructured and vary significantly in format, necessitating intelligent systems capable of extracting and understanding the essential details.

This task assesses your ability to effectively combine vision, language processing, and structured reasoning into a coherent and practical solution. Your goal is to create a small service capable of accepting uploaded claim sheet images, extracting structured data, and responding to questions about that data.

Task Objective

Develop a Python microservice using either **FastAPI** or **Flask** that includes two key endpoints:

1. `POST /extract`

- **Input**: Image or PDF file containing a medical claim sheet.
- **Output**: Structured JSON object with key details extracted from the document.

An example output structure could be like this:

```
{
  "patient": {
    "name": "Jane Doe",
    "age": 34
  },
  "diagnoses": ["Malaria"],
  "medications": [
    {"name": "Paracetamol", "dosage": "500mg", "quantity": "10 tablets"}
  ],
  "procedures": ["Malaria test"],
  "admission": {
```

```
    "was_admitted": true,
    "admission_date": "2023-06-10",
    "discharge_date": "2023-06-12"
  },
  "total_amount": "₦15,000"
}
```


2. `POST /ask`

- **Input**:


```
{
  "document_id": "abc123",
  "question": "How many tablets of paracetamol were prescribed?"
}
```


- **Output**:


```
{
  "answer": "10 tablets"
}
```


Guidelines

- You can choose any OCR approach that suits your implementation:
    - Traditional OCR libraries
    - Vision-capable LLMs (such as Gemini Vision, GPT-4 Vision, etc.)
- You are encouraged to use free-tier versions of LLMs available (e.g., Google's Gemini or other freely accessible models).
- In-memory storage is sufficient to retain extracted data for subsequent questions.

Resources

Here is a link to a Google Drive folder containing sample images:

- https://drive.google.com/drive/folders/1Zl8spNwE7xe8jMaG3GUilwvkE0z73-e4?usp=sharing

Submission Instructions

- Push your completed solution to a **public GitHub repository**.
- Include a detailed `README.md` covering:
    - Your overall approach
    - Assumptions and decisions made during implementation
    - Instructions on how to run your service locally, including dependencies
- Provide the GitHub repository link upon submission.

Expectations

- This task should be completed within **2–3 days**.
- Consider reasonable architectural decisions around extensibility, maintainability and good structure.

Evaluation Criteria

- Cleanliness, readability, and structural clarity of your code
- Creativity and effectiveness in extracting and reasoning with data
- Thoughtfulness in the use of LLMs and other tools.
- Practical engineering considerations such as scalability, and maintainability.
- Clarity and completeness of your README documentation

We look forward to reviewing your submission.