Project Title: AI-Powered Marksheet Extraction API

Name: Gul Mittal

Date: January 22, 2026

1. Problem Statement

The end objective here was to develop an efficient extraction API to parse unstructured academic results (PDFs/Images) into structured JSON. The major problem here lies in dealing with different layouts such as grid layouts and merged columns, as well as varying image quality.

2. Core Approach: Multimodal LLM vs. Traditional OCR

Instead of utilizing a regular OCR technique (like Tesseract -> Regex Parsing), I have utilized a Multimodal LLM technique (like Gemini 1.5 Flash).

Why? Because traditional methods lose spatial information. Marksheets often involve a lot of grid-related positioning. "Maths" often needs to be lined up with "85". Because faint gridlines tend to confound Tesseract.

Solution: Gemini 1.5 Flash directly processes the pixels of the images. It "sees" the row and column relationships and can therefore correctly interpret complex tables and grouped subjects such as "Language Group."

3. Technical Architecture

- Preprocessing: The pdf2image library converts the PDF into image form, thereby providing the LLM with an established visual structure.
- Validation: The Pydantic model used for LLM tasks strictly adheres to an implied JSON schema. This guarantees that while LLM generation may fluctuate, API response structure will remain consistent, for instance, in terms of float-type requirement for marks.
- Backend: Built using the fastapi library to utilize the concurrency benefits provided by asynchronous execution for multiple requests.

4. Confidence Score Logic (Reflective Scoring)

As raw probabilities of individual tokens are often not obtainable, I've also incorporated a Reflective Confidence Mechanism.

- Method: In the system prompt, there are explicit instructions to the models to assess the visual legibility of the document.
- Scoring Criteria:
    - 0.95 - 1.0: Crystal clear digital text/high res scan.
    - 0.80 - 0.90: Slight skew, minor blur, or handwriting.

- < 0.80: Significant artifacts, watermarks, and unclear column alignment.

5. Key Design Choices

- Dockerization: The project includes a Dockerfile to manage system-level dependencies like poppler-utils so that API effectively runs across every cloud platform.
- Modular Code : The project's modular structure means the code for the program's logic is neatly separated and organized within the following files and modules: services.py (for AI), utils.py (Image processing), and schemas.py (Data definition)