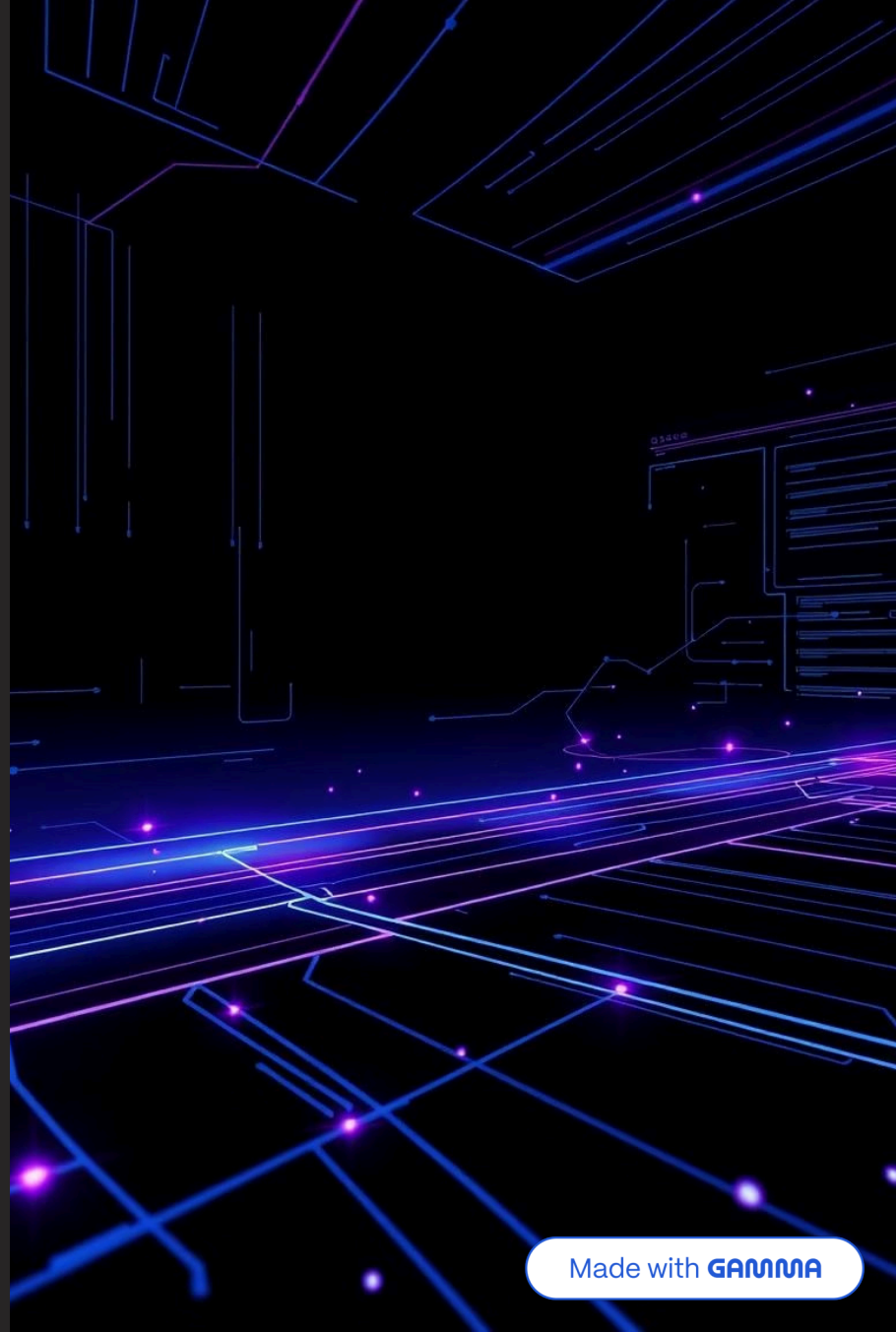
Proper PDF Extractor

Adobe Hackathon 2025 Submission (Challenge 1A)

A high-precision PDF document structure extraction engine designed for the Adobe India Hackathon 2025, delivering unparalleled accuracy and performance.



Project Overview

This project offers an innovative, high-precision solution for extracting structured outlines from PDF documents, including hierarchical headings and their corresponding page numbers.

Offline Operation

Engineered to operate entirely offline, ensuring data security and continuous functionality without internet dependency.

Minimal Footprint

Designed for efficiency with a minimal resource footprint, making it suitable for diverse environments.

Exceptional Speed

Delivers extraction results with remarkable speed, optimizing workflow efficiency.

Core Methodology: Human-First Approach

Our solution translates human intuition into a highly efficient, rule-based algorithm, mirroring how a person would manually identify document structure.

Deep Dive & Deconstruction

Thoroughly studied sample PDFs to understand nuances and precise output structures.

Ideation Phase

Deconstructed problems, brainstormed features, and mind-mapped structural identification from document features.

Automating Intuition

Programmatically implemented human workflow steps: title identification, style analysis, and pattern recognition.

Guiding Principles: Mimicking Human Workflow

Our script programmatically implements the intuitive steps a human takes when identifying document structure.

1 **Title Identification**

First, it identifies the document's title, typically at the start or in metadata.

2 **Style Analysis**

Recognizes larger, bolder, or distinctively formatted headings.

3 **Pattern Recognition**

Identifies structural markers like "Chapter 1" or "1.1."

4 **Contextual Filtering**

Ignores non-essential elements such as page numbers, footers, and dates.

5 **Positional Heuristics**

Applies rules that prevent headings from being identified in illogical positions, like the very bottom of a page.

Technology Stack and Rationale

Each tool was meticulously selected based on official documentation to meet performance and constraint requirements.

PDF Text Extraction	PyMuPDF (fitz)	Unmatched speed, rich low-level data (font size, weight, bounding boxes).	pdfplumber (slower), PyPDF2/4 (lacks detail), PDFMiner (complex/slower).
AI / Intelligence	None (Deliberate Choice)	Guarantees maximum speed, 100% deterministic results, zero-byte footprint.	ML models (size, speed, ambiguity overhead).
Pattern Matching	Regular Expressions (re)	Powerful and efficient for filtering and identifying heading styles.	No practical alternatives in standard Python library.
File/Data Handling	os, json	Standard, reliable Python libraries for paths and JSON output.	pathlib (sufficient functionality with os and json).

Multilingual Support

A key enhancement: our extractor seamlessly handles documents in multiple languages, ensuring robust performance across diverse global datasets without requiring language-specific customization.

Solution Architecture & Workflow

The **app.py** script orchestrates the batch processing of all documents, ensuring efficient and robust operation.

Initialization

Identifies **/app/input** and **/app/output** directories within the container.

File Iteration

Iterates through every **.pdf** file in the input directory.

Instantiation & Execution

Creates a new **ProperPDFExtractor** instance for each PDF, ensuring a clean state, and calls **extract_outline()**.

Output Generation

Serializes the resulting dictionary into a human-readable JSON file and saves it to the output directory.

Error Handling

Ensures a single corrupted PDF does not halt the entire batch operation via a **try...except** block.

Setup and Execution Instructions

The solution is containerized using Docker, providing a sandboxed and reproducible environment. No host-machine setup is required beyond Docker installation.

Build the Docker Image

The Dockerfile installs all dependencies during the build step; no internet connection is required during runtime.

```
docker build --platform linux/amd64 -t pdf-processor .
```

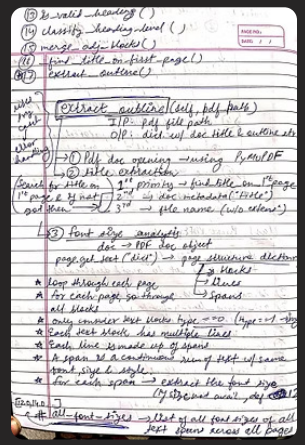
Run the Analysis

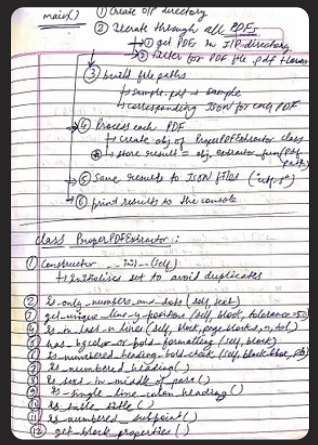
This command processes all PDFs in your input folder and places the results in your output folder.

```
docker run --rm -v $(pwd)/input:/app/input:ro -v $(pwd)/output:/app/output --network none pdf-processor
```

Our Ideation Process: From Notes to Code

Our algorithm is a direct result of our manual brainstorming and mind-mapping, forming the foundational logic of our code.





Key Takeaways & Next Steps

Highlights of Proper PDF Extractor:

- High-precision, rule-based extraction.
- Offline capability and minimal resource footprint.
- Exceptional speed and multilingual support.
- Human-first analytical approach to document understanding.

Future Enhancements:

- Expand to include table of content (TOC) extraction.
- Integrate advanced layout analysis for complex documents.
- Develop a user-friendly GUI for easier interaction.
- Explore cloud deployment options for broader accessibility.

Thank you for your time. We welcome your questions!