Welcome to the “Connecting the Dots” Challenge

Rethink Reading. Rediscover Knowledge

What if every time you opened a PDF, it didn’t just *sit there*—it *spoke to you*, *connected ideas*, and *narrated meaning* across your entire library?

That’s the future we’re building — and we want **you** to help shape it.

In the **Connecting the Dots Challenge**, your mission is to reimagine the humble PDF as an intelligent, interactive experience—one that **understands structure**, **surfaces insights**, and responds **to you** like a trusted research companion.

The Journey Ahead

- **Round 1:**

  Kick things off by building the brains — extract structured outlines from raw PDFs with blazing speed and pinpoint accuracy. Then, power it up with on-device intelligence that understands sections and links related ideas together.

- **Round 2:**

  It’s showtime! Build a beautiful, intuitive reading webapp using Adobe’s PDF Embed API. You will be using your Round 1 work to design a futuristic webapp.

Why This Matters

In a world flooded with documents, what wins is not more content — it’s **context**. You’re not just building tools — you’re building the future of how we **read, learn, and connect**. No matter your background — ML hacker, UI builder, or insight whisperer — this is your stage.

Are you in?

It’s time to read between the lines. Connect the dots. And build a PDF experience that feels like **magic**. Let’s go.

Round 1A: Understand Your Document

Challenge Theme: Connecting the Dots Through Docs

Your Mission

You're handed a PDF — but instead of simply reading it, you're tasked with making sense of it like a machine would. Your job is to extract a structured outline of the document — essentially the **Title**, and headings like **H1**, **H2**, and **H3** — in a clean, hierarchical format.

This outline will be the **foundation** for the rest of your hackathon journey.

Why This Matters

PDFs are everywhere — but machines don’t naturally understand their structure. By building an outline extractor, you’re enabling smarter document experiences, like semantic search, recommendation systems, and insight generation.

What You Need to Build

You must build a solution that:

- Accepts a **PDF file** (up to 50 pages)
- Extracts:
  - **Title**
  - **Headings:** H1, H2, H3 (with level and page number)
- Outputs a valid JSON file in the format below:

```
{
 "title": "Understanding AI",
 "outline": [
  { "level": "H1", "text": "Introduction", "page": 1 },
  { "level": "H2", "text": "What is AI?", "page": 2 },
  { "level": "H3", "text": "History of AI", "page": 3 }
 ]
}
```

You Will Be Provided

1. A **sample input PDF** (e.g., sample.pdf)
2. A **sample ground truth output** (sample.json) for format clarity

3. Sample Dockerfile
4. Sample Solution

Docker Requirements

-
- Please ensure your Dockerfile is compatible with AMD64 architecture. Since we will build and run the image on an AMD64 machine, your base image and any dependencies should support linux/amd64. Optionally, you can include the following in your Dockerfile to explicitly specify the platform: FROM --platform=linux/amd64 <base_image>
- **CPU architecture:** amd64 (x86_64)
- No GPU dependencies
- Model size (if used) ≤ 200MB
- Should work offline — no network/internet calls

Expected Execution

We will build the docker image using the following command:

```docker build --platform linux/amd64 -t mysolutionname:somerandomidentifier```

After building the image, we will run the solution using the run command specified in the submitted instructions.

```docker run --rm -v $(pwd)/input:/app/input -v $(pwd)/output:/app/output --network none mysolutionname:somerandomidentifier```

Your container should:

- Automatically process all PDFs from /app/input directory, generating a corresponding filename.json in /app/output for each filename.pdf
- output.json

Constraints

Constraint	Requirement
Execution time	≤ 10 seconds for a 50-page PDF

Model size	≤ 200MB (if used)
Network	**No internet access allowed**
Runtime	Must run on CPU (amd64), your solution should run on the system with 8 CPUs and 16 GB RAM configurations

Scoring Criteria

Criteria	**Max Points**
Heading Detection Accuracy (Precision + Recall)	25
Performance (Time & Size Compliance)	10
Bonus: Multilingual Handling (e.g., Japanese)	10
Total	45

Submission Checklist

1. Git Project with a working Dockerfile in the root director and
2. A working **Dockerfile**
3. All dependencies installed within the container
4. A README.md that explains:
    - Your approach
    - Any models or libraries used
    - How to build and run your solution (This is purely for documentation purpose, your solution should run using the “Expected Execution” section above.

Pro Tips

- Don’t rely solely on font sizes for heading level determination — headings in some PDFs break that assumption.
- Test your solution across both simple and complex PDFs.
- Make your code modular — you’ll reuse this structure in Round 1B.

- Important – Please keep your Git Repo private till the competition deadline, you will be informed, when to make the repo public.

What Not to Do

- Do not hardcode headings or file-specific logic
- Do not make API or web calls
- Do not exceed the runtime/model size constraints

[[Public Dataset Folder]]

(For Sample Input and Output Files, please refer to the appendix)

Round 1B: Persona-Driven Document Intelligence

Theme: “Connect What Matters — For the User Who Matters”

Challenge Brief (For Participants)

You will build a system that acts as an intelligent document analyst, extracting and prioritizing the most relevant sections from a collection of documents based on a specific persona and their job-to-be-done.

Input Specification

2. **Document Collection**: 3-10 related PDFs **Persona Definition**: Role description with specific expertise and focus areas
    3. **Job-to-be-Done**: Concrete task the persona needs to accomplish

Document collection, persona and job-to-be-done can be very diverse. So, the solution that teams need to build needs to be generic to generalize to this variety.

- Documents can be from any domain (Example: Research papers, school/college books, financial reports, news articles etc.)
- Persona can again be very diverse (Example: Researcher, Student, Salesperson, Journalist, Entrepreneur etc)
- Job-to-be-done: This will be related to the persona (Example: Provide a literature review for a given topic and available research papers, What should I study for Organic Chemistry given the chemistry documents, Summarize the financials of corporation xyz given the detailed year end financial reports etc.)

Sample Test Cases

Test Case 1: Academic Research

- **Documents**: 4 research papers on "Graph Neural Networks for Drug Discovery"
- **Persona**: PhD Researcher in Computational Biology
- **Job**: "Prepare a comprehensive literature review focusing on methodologies, datasets, and performance benchmarks"

Test Case 2: Business Analysis

- **Documents**: 3 annual reports from competing tech companies (2022-2024)
- **Persona**: Investment Analyst
- **Job**: "Analyze revenue trends, R&D investments, and market positioning strategies"

Test Case 3: Educational Content

- **Documents**: 5 chapters from organic chemistry textbooks
- **Persona**: Undergraduate Chemistry Student
- **Job**: "Identify key concepts and mechanisms for exam preparation on reaction kinetics"

Required Output

- Output JSON format: Refer *challenge1b_output.json*

The output should contain:

1. Metadata:
   a. Input documents
   b. Persona
   c. Job to be done
   d. Processing timestamp
2. Extracted Section:
   a. Document
   b. Page number
   c. Section title
   d. Importance_rank
3. Sub-section Analysis:
   a. Document
   b.
   c. Refined Text
   d. Page Number **Constraints**

- Must run **on CPU** only
- Model size **≤ 1GB**
- Processing time ≤ 60 **seconds** for document collection (3-5 documents)
- **No internet access** allowed during execution

Deliverables

- approach_explanation.md (300-500 words explaining methodology)
- Dockerfile and execution instructions

Sample input/output for testing

-

Scoring Criteria

Criteria	Max Points	Description
Section Relevance	60	How well selected sections match persona + job requirements with proper stack ranking
Sub-Section Relevance	40	Quality of granular subsection extraction and ranking

Appendix:

https://github.com/jhaaj08/Adobe-India-Hackathon25.git