Adobe India Hackathon 2025 – Grand Finale

Connecting the Dots Challenge

Theme: From Brains to Experience – Make It Real

Introduction

Congratulations on making it to the Finale!

In Rounds 1A & 1B, you built:
- Round 1A – A robust PDF understanding engine.
- Round 1B – A persona-driven document intelligence system.

Now, it's time to turn those "brains" into a real, interactive user experience.

User Journey – Document Insight & Engagement System

1. Context & Problem

Users (e.g., researchers, students, professionals) deal with large volumes of documents daily — research papers, business reports, study material, etc. Over time, it becomes impossible to remember all details or connect insights across these documents.

2. Goal of the System

Help users by:

- Quickly surfacing related, overlapping, contradicting, examples or other insightful information from their personal document library.
- Using AI/LLM-powered capabilities to enhance understanding and engagement — grounded on the documents they've read.

3. Journey Flow

Step 1 – Reading & Selection

- **Trigger:** User is reading a document within the system.
- **Action:** User selects a portion of text (e.g., a scientific method, a business strategy, a key metric).

- **System Response:** Instantly surfaces relevant sections from past documents in the user's library. Uses semantic search and optionally an LLM to ensure context-aware matching. Speed and quality is important for user engagement.

Step 2 – Insight Generation

- **Goal:** Go beyond finding related text.
- **System adds value by:** Generate **insights** related to the selected text. E.g. overlapping, contradictory viewpoints, examples, or other insights. Offering contextual insights that enrich understanding. Grounding all results in documents the user has actually read (uploaded) — not generic web sources.

Step 3 – Rich Media Experience

- **Optional Action:** User requests an audio overview / podcast for the selected topic.
- **System Capabilities:** Generate a natural-sounding, engaging audio overview of the topic using the selected text as a seed. Pull content from user's documents to maintain trust and accuracy. Structure audio for easy listening — highlights key points, contrasts perspectives, and connects concepts. Making the audio overview/podcast natural sounding, engaging, contextual, and grounded is key.

4. Key UX Considerations

- **Speed:** Minimize delay between text selection and insight surfacing — keeps user engaged.
- **Relevance:** High-relevance matches ensure trust in the system.
- **Engagement:** Audio should be natural and dynamic, not robotic.
- **Extensibility:** Users can explore beyond core tasks (bonus features) while staying aligned with the main flow.

5. Example Use Case

A researcher reading a paper on "neural network training techniques" selects a paragraph on "transfer learning." The system instantly shows:

- Similar methods in 3 previous papers.
- Contradictory findings from another study. Or how another paper has extended the technique. Or how another paper has found problems with the technique.
- An audio overview / podcast summarizing these related sections and insights for quick listening on the go.

Your Mission

Build a web-based PDF reading experience powered by your earlier work that:
1. Displays PDFs with full fidelity & zoom/pan.
2. Connects the dots — when the user selects some text, shows related sections and snippets from other PDFs.
3. Accelerates understanding by showing context-aware insights.
4. Adds Audio Overview / Podcast features for richer engagement.

Core Features (Mandatory)

PDF Handling

- Bulk Upload: User uploads multiple PDFs at once (represents "past documents" that the user have read).
- Fresh Upload: User opens an additional new PDF (represents "current document" that the user is reading).
- Display: Render PDFs at high fidelity (using PDF Embed API is preferred).

Connecting the Dots

- Identify & highlight upto 5 relevant sections across PDFs with high accuracy.
- "Sections" = Just as was defined in Round 1A, the headings in a PDF logically break the PDF into a number of sections (heading along with their content)
- "Snippets" = 2–4 sentence extracts from the context of the section. Similar to how web-search results how relevant snippets along with the URL + Title.
- When the user clicks a snippet, show the corresponding PDF and navigate to the relevant part of the PDF containing that section.

Speed

- Related sections/snippets must load quickly after selection for better user engagement.
- Ingestion speed for past documents follows earlier round limits.

Follow-On Features (Optional, Bonus Points)

Insights Bulb (+5 points)

Insights can use LLM calls. Some kinds of insights could be:
- Key takeaways
- "Did you know?" facts
- Contradictions / counterpoints
- Examples

- Cross-document inspirations

Audio Overview / Podcast Mode (+5 points)

- 2-5 min audio podcast (between 2 speakers) or audio overview(single speaker). Audio podcast is preferred but audio overviews are also acceptable. It should be based on:
  - Current section

  - Related sections
  - Insights from Bulb feature
- Azure TTS for evaluation (Google TTS/local allowed for development).

Clarifications from Jury Q&A

- Snippets logic: Select text → system finds relevant sections/snippets → user clicks to jump.
- Connections: Automatic linking across PDFs based on semantic meaning (it will be judged for relevance).
- Prompt bar: Not required; use context from selection (and surrounding text if needed)/
- LLM Usage: API calls allowed; Round 1B logic can be replaced.
- Offline requirement: Only LLM, TTS, and Embed API may use internet.
- Backend: Fully runnable in Docker (combined frontend + backend).
- Performance: No strict execution limit; faster is better.
- Model size: Preferably under 20 GB for Docker image size.

How will your solution be built and run.

The following docker command will be used to build the solution.

docker build --platform linux/amd64 -t yourimageidentifier .

The following docker command will be used to run your solution

docker run –v /path/to/credentials:/credentials -e ADOBE_EMBED_API_KEY=<ADOBE_EMBED_API_KEY> -e LLM_PROVIDER=gemini -e GOOGLE_APPLICATION_CREDENTIALS=/credentials/adbe-gcp.json -e GEMINI_MODEL=gemini-2.5-flash -e TTS_PROVIDER=azure -e AZURE_TTS_KEY=TTS_KEY -e AZURE_TTS_ENDPOINT=TTS_ENDPOINT -p 8080:8080 yourimageidentifier

Running the above command should bring up an application accessible on http://localhost:8080

Note that making external LLM/TTS calls is entirely optional, and candidates may choose any on-device solution(like ollama for LLM or festival for TTS) which does not make any external calls. However, if any external LLM/TTS calls are required, they must use the environment variables(which will be passed by Adobe). This is to ensure that user credentials/API keys are not embedded inside the code in git repo.

Sample Scripts

- LLM call: https://github.com/rbabbar-adobe/sample-repo/blob/main/chat_with_llm.py
- Generate Audio from text: https://github.com/rbabbar-adobe/sample-repo/blob/main/generate_audio.py

- Dependencies to be installed for the above: https://github.com/rbabbar-adobe/sample-repo/blob/main/requirements.txt

Note that the generate_audio.py script generates mp3 file for a text. Multiple invocations of the script may be required to generate an audio podcast involving multiple speakers. For audio overviews(with a single speaker), a single invocation may be enough.

Candidates are expected to include the above scripts in their code and make use of the their functions to make LLM/TTS calls so that their solution work as expected in the evaluation. If the solution involves a non-python based solution, then a similar script in the other language may be used, if it respects the environment variables.

Note that the evaluation will be done using 'Gemini-2.5-Flash' for LLM, and Azure TTS. However, candidates may use other LLM/TTS solutions which they have access to for local development and validation, as they are supported by the scripts above.

Here are the list of environment variables along with the details

Environment Variables which will be passed during evaluation.

Environment Variable	Value	Description
ADOBE_EMBED_API_KEY	To be provided by the candidate.	Optional, If you are using Adobe PDF Embed API, then this should be provided by the candidates while submitting their solution.
LLM_PROVIDER	gemini	
GOOGLE_APPLICATION_CREDENTIALS	Will be set by Adobe.	
GEMINI_MODEL	gemini-2.5-flash	
TTS_PROVIDER	azure	
AZURE_TTS_KEY	Will be set by Adobe.	
AZURE_TTS_ENDPOINT	Will be set by Adobe.	

The sample scripts shared above provide the ability to make use of other LLM/TTS providers. See some possible examples below.

Using Gemini for both LLM and TTS calls.

docker run –v /path/to/credentials:/credentials -e ADOBE_EMBED_API_KEY=<ADOBE_EMBED_API_KEY> -e LLM_PROVIDER=gemini -e GOOGLE_APPLICATION_CREDENTIALS=/credentials/adbe-gcp.json -e GEMINI_MODEL=gemini-2.5-flash -e TTS_PROVIDER=gcp -p 8080:8080 yourimageidentifier

Using Local LLM(using Ollama) and local TTS implementation.

docker run -e ADOBE_EMBED_API_KEY=<ADOBE_EMBED_API_KEY> -e LLM_PROVIDER=ollama -e OLLAMA_MODEL =llama3 -e TTS_PROVIDER=local -p 8080:8080 yourimageidentifier

Deliverables

Submit by deadlines:
1. Working Prototype (Docker runnable, accessible on localhost:8080).
2. Private GitHub Repo with code, Dockerfile, README.
3. Pitch Deck (max 6 slides): Problem, solution, features, innovation, demo, impact.
4. 2-min demo video.
5. ADOBE_EMBED_API_KEY(optional, if using Adobe Embed API): to be submitted via form.

Evaluation Criteria

Stage 1 – Backend Evaluation (50%)

Core Functionality – 20 points
Technical Implementation – 15 points
Integration of Prior Rounds – 10 points
Performance & Reliability – 5 points

Stage 2 – Live Finale (50%)

Demo Effectiveness – 15 points
UX & Design Quality – 10 points
Innovation & Creativity – 10 points
Impact & Storytelling – 10 points
Q&A Handling – 5 points

Bonus: +5 for Insights Bulb, +5 for Podcast Mode

8. Key Dates

- Code Freeze: 19 Aug 2025
- Pitch Deck Deadline: 3 Sep 2025
- Finale Day: 5 Sep 2025

9. Pro Tips

- Tell a story – show real persona impact.
- Highlight “magic moments” in your demo.
- Prepare metrics showing time saved / comprehension improved.
- Rehearse until it’s smooth & predictable.