

EJUST APPATHON - Ideation Log

Ideation

Prompt 1

1. Input:

```
<date>Aug 2</date>

<model>
gpt-o3
</model>

<prompt>
Help me brainstorm tech stack, as well as architecture for the following specs. Use is an
appathon. so we do not need something very complex, just a working prototype. Below are my
initial thoughts and brainstorming:

- We need to create a personalized study tool
- RAG (multi-modal. I am thinking of using R2R as it proved to be of high performance) to take
hand-written notes (OCR)
- Add ability to read diagrams, charts, circuits.
- Add ability to recreate documents. [think about]

----

## Specs

#### Definition of the solution:
- Web app using flask, firebase, and react.
- Integrate ocr enginer (tesseract, ocr 2.0, or build our own?)
- Integrate r2r. How to manage hosting? Docker.
</prompt>
```

2. output(truncated):

Layer	Quick-win choice	Why it works for a hackathon
Frontend	React + Vite, TypeScript, Tailwind	Fast scaffolding, hot-reload; Vite bundles in milliseconds.
State & Auth	Firestore (Auth, Firestore, Storage)	Zero-config auth + real-time DB; generous free tier.
Backend API	Flask (REST & WebSocket endpoints)	Few-file codebase; easy to dockerize.
Retrieval system	R2R-Light running as its own service	Multimodal ingestion, hybrid search, Graph-RAG, REST API; drop-in Docker image. GitHubPyPlr2r-docs.sciphi.ai
Vector DB	Built-in pgvector inside R2R-Light or external Qdrant	R2R ships with adapters; pgvector keeps ops minimal.
OCR pipeline	pytesseract for plain text plus**PaddleOCR** or opencv-python for diagrams/charts	PaddleOCR handles mixed layouts better; both are pure-Python.
LLM for generation	OpenAI GPT-4o (hosted) or local Llama-3-8B via Ollama if offline	R2R can call either via its “Deep Research” agent.

Layer	Quick-win choice	Why it works for a hackathon
Container orchestration	Docker Compose (3-service stack)	Single YAML, easy to demo on any laptop.

- **Upload flow**

- User drops a handwritten photo or PDF.
- Flask streams the file to **Firestore Storage** and triggers the **OCR service**.
- OCR returns JSON (`{"text": ..., "svg": ...}`) that Flask forwards to R2R's `/documents.create`.
- R2R chunks, embeds, and indexes the text + image embeddings.

- **Chat / Q&A**

- Frontend sends a study question → Flask hits `R2R /rag` → R2R returns passages + citations → Flask calls the LLM with the retrieved context → answer streams to the UI.

- **Diagram understanding**

- Diagrams are fed through PaddleOCR's table/figure detector; extracted captions + the raw image are stored.
- During retrieval, R2R's **hybrid search** matches either text or CLIP embeddings for the image, giving you multimodal hits. [arXiv](#)

- **Document recreation**

- A "Re-create" button calls an LLM prompt that formats the retrieved notes into markdown → `python-docx` or `pdfkit` turns it into a DOCX/PDF on the fly → returns downloadable link.

We asked gpt-o3 to create a mermaid of the architecture below:



