RAG-GP Project – Phase 2 Instructions (Weeks 1–4)

Objective

Advance from a multi-format retriever prototype to a functional CLI-based RAG system capable of producing LLM-generated, context-aware answers over a comprehensive Alma corpus — including image-aware and XLSX-aware retrieval paths.

☑ Current Status (as of now)

- ☑ Ingestion pipeline supports: **PDF, DOCX, PPTX, TXT, XLSX**
- ☑ Chunking rules applied per format; FAISS indices built per type
- ☑ Local and OpenAI embedding modes fully functional
- ☑ Retrieval pipeline operational with **late fusion**, **per-type recall**, and **query embedding** via `text-embedding-3-large`
- ☑ CLI command `retrieve` works across mixed-type corpus with debug output
- ☑ Retrieval metadata preserved (e.g., `doc_id`, `doc_type`, etc.)

🗓 4-Week Plan (Phase 2: Retrieval→Answer MVP)

Week	Focus Area	Tasks	Output
1	**Corpus Expansion**	- Collect 50–100 Alma documents across key modules (Acquisitions, Analytics, Fulfillment, etc.) - Index by format and theme - Store in `input/` per project	Expanded `input/`, organized by topic and format
1	**Image-Aware Setup**	- In PDF/DOCX chunkers: detect embedded images - Extract them to disk/cache dir - Bind to parent chunk ID/location	Images linked to chunk metadata (`image_path`, `image_page`)
2	**OCR Extraction for Screenshots**	- Run OCR via `tesserocr` or `pytesseract` - Store text in `image_ocr_text` field of chunk metadata	OCR data stored alongside chunk content
2	**Evaluate Retriever at Scale**	- Run `retrieve` CLI over new corpus - Save top-K results and log chunk source types	Retrieval report, sample queries with diverse results

Week	Focus Area	Tasks	Output
2	**LLM Prompt Template**	- Draft first prompt format: QA with citations using top-K chunks - Optionally inject OCR text	`prompt_builder.py` base implementation
3	**Implement `ask()` CLI Endpoint**	- Add CLI: `ask <project_path> <query>` - Retrieves top-K chunks, formats prompt, calls OpenAI/GPT	End-to-end CLI answer generation
3	**Image-Inclusive Answering**	- Concatenate `chunk.text + image_ocr_text` for embedding + prompt inclusion - Format screenshots descriptively if OCR present	Answers influenced by visual context
4	**Evaluate Answer Quality**	- Manual analysis of 10–20 queries - Label precision, hallucination, citation quality - Mark XLSX or multi-source failure cases	Annotated QA sample set, metrics summary
4	**Begin XLSX-Aware Chunker**	- Refactor chunker to process per-table - Capture headers, rows, and normalize content - Optionally summarize table intent	`XLSXChunkerV2`, structured chunks with field-level clarity
4	**Start Agent Hub Skeleton**	- Build `AgentHub` interface with hooks - Add `RerankerAgent` (e.g., re-sort chunks by alignment score) - Plan `SynthesizerAgent` (cross-chunk integration)	Agent interface and minimal reranker stub

🧱 Component Ownership and Notes

Component	Owner / Task Lead	Location / Notes
Corpus curation	You	Organize under `data/projects/demo_project_full/`
OCR integration	Chunker maintainer	Can reuse image utils from PDF chunker, extend to DOCX
Prompt templates	`prompt_builder.py`	Start with simple QA+source format; plan for citation markup
CLI commands	`app/cli.py`	`ask()` should mirror `retrieve()` in logging/debug output
LLM gateway	`scripts/llm/gateway.py`	First version can assume OpenAI; later make model pluggable

Component	Owner / Task Lead	Location / Notes
Agent Hub	`scripts/agents/hub.py`, registry pattern	Each agent gets access to query, chunks, and config

🧠 Optional Enhancements (if ahead of schedule)

Task	Description
GPT-4o or Gemini captioning	Generate natural-language captions for key screenshots
Table-to-text converter	Generate per-table descriptions from XLSX sheets
Fusion analysis script	Visualize score distribution across doc types
Streamlit debug viewer	Visualize retrieval + chunks + scores per query

📌 Immediate Next Steps (Week 1 Focus)

1. **Organize and load Alma instructional documents (target: 50–100)**
2. **Enable screenshot extraction + OCR from PDFs and DOCX**
3. **Log and verify image/ocr attachment per chunk**
4. **Prepare 5–10 pilot queries for round-trip `retrieve()` test**