RAG Knowledge Base — 28-Day Daily Task Sheet

Purpose: a day-by-day checklist you can upload to NotebookLM. Each day: Build → Verify → Commit. Time boxes assume Wed/Fri are deep-focus days.

How to use

1) Do the Build items. 2) Run the Verify step exactly. 3) Commit with the suggested message. If you miss a day, do not skip verification.

Week 1 · Walking Skeleton → Ingestion → RAG MVP

Day	Focus	Build	Verify	Commit
Day 1	Backend skeleton	Create FastAPI app + <b>/health</b> + <b>/chat</b> (mock).	Open <b>/docs</b>; test POST <b>/chat</b> returns mock JSON.	git commit -m "Day1 backend skeleton"
Day 2	Frontend skeleton	Create Streamlit chat UI calling backend <b>/chat</b>.	Streamlit shows backend response; handles connection errors.	git commit -m "Day2 frontend skeleton"
Day 3	PDF ingestion	Add POST <b>/upload</b> + service <i>load_and_chunk_pdf</i> (PyPDFLoader + splitter).	Upload a PDF; return <i>num_chunks</i> + sample chunk metadata.	git commit -m "Day3 ingestion + chunking"
Day 4	Embeddings + FAISS	Build FAISS index from chunks using embeddings; persist locally.	Build index; verify vector count equals chunks count.	git commit -m "Day4 embeddings + FAISS"
Day 5	Retrieval-only API	Add <b>/search</b>: query -> top_k chunks (no LLM yet).	Query returns relevant chunk texts + metadata.	git commit -m "Day5 retrieval endpoint"
Day 6	RAG answer MVP	Update <b>/chat</b>: retrieve -> LLM answer (minimal prompt).	Ask question; answer uses retrieved context (spot-check).	git commit -m "Day6 RAG answer MVP"
Day 7	Weekly harden	Refactor folders; add .env config; add basic logging.	Run backend + frontend end-to-end after refactor.	git commit -m "Week1 wrap"

Notes (fill in after you finish the week): What broke? What did you learn? What will you simplify next week?

Week 2 · Quality: citations, memory, evaluation, docs

Day	Focus	Build	Verify	Commit
Day 8	Citations	Return source docs (page/chunk) in API response.	Answer includes a sources list with page numbers.	git commit -m "Day8 citations"
Day 9	UI citations	Render citations in Streamlit under each answer.	Citations readable; no crash if missing.	git commit -m "Day9 UI citations"
Day 10	Memory (lite)	Store last 3 turns in session_state; send to backend.	Follow-up question resolves references (this/that).	git commit -m "Day10 chat history"
Day 11	Chunk tuning	Make chunk params configurable; log chunk stats.	Compare 2 settings; note quality trade-offs.	git commit -m "Day11 chunk config"
Day 12	Retrieval quality	Add top_k control + optional scores; add tiny eval notes.	Run 10 queries; record quick hit-rate notes.	git commit -m "Day12 retrieval eval"
Day 13	Error handling	Handle bad PDFs, empty queries, missing index; friendly errors.	Try failure cases; UI shows clear messages.	git commit -m "Day13 error handling"
Day 14	Docs + diagram	Write README (setup/run) + simple architecture diagram.	Fresh clone: setup steps succeed end-to-end.	git commit -m "Week2 wrap"

Notes (fill in after you finish the week): What broke? What did you learn? What will you simplify next week?

Week 3 · Productization: Docker + Deployment + Demo

Day	Focus	Build	Verify	Commit
Day 15	Docker backend	Dockerfile for backend; run locally.	docker build/run; /docs reachable.	git commit -m "Day15 docker backend"
Day 16	Docker frontend	Dockerfile for frontend + docker-compose for both.	docker-compose up runs both services.	git commit -m "Day16 docker compose"
Day 17	Deploy prep	Pick host; define start commands; document env vars.	Local config mirrors deployment commands.	git commit -m "Day17 deploy prep"
Day 18	Deploy backend	Deploy backend; verify public <b>/health</b>.	Public /health returns ok.	git commit -m "Day18 deploy backend"
Day 19	Deploy frontend	Deploy Streamlit; point to backend URL.	Public UI loads; can chat successfully.	git commit -m "Day19 deploy frontend"
Day 20	Observability	Add request logging + basic timing (retrieval/LLM).	Logs show latency per request.	git commit -m "Day20 observability"
Day 21	Demo ready	Add sample PDF + demo questions; polish UX.	A friend can use it from README alone.	git commit -m "Week3 wrap"

Notes (fill in after you finish the week): What broke? What did you learn? What will you simplify next week?

Week 4 · Polish: reliability + resume packaging

Day	Focus	Build	Verify	Commit
Day 22	Prompt + fallback	Add strict 'use context only' prompt + no-context fallback.	When no context, model says it cannot answer.	git commit -m "Day22 prompt"
Day 23	Security basics	Limit upload size; sanitize filenames; keep secrets out of repo.	Large file rejected; no secrets committed.	git commit -m "Day23 security"
Day 24	Caching	Cache index per file; avoid rebuilding every chat.	Second query faster; cache-hit logged.	git commit -m "Day24 caching"
Day 25	Testing	Add minimal pytest for ingestion + search.	pytest passes locally.	git commit -m "Day25 tests"
Day 26	Resume package	Write 3 resume bullets + 2-min project story.	You can explain trade-offs clearly.	git commit -m "Day26 resume"
Day 27	Mock (project)	Do 1 project walk-through mock; capture Q&A.	Notion updated with Q&A + gaps.	git commit -m "Day27 mock"
Day 28	Final ship	Tag v1.0; verify demo URL + screenshots + docs.	Everything works from README; share URL.	git commit -m "Day28 v1.0"

Notes (fill in after you finish the week): What broke? What did you learn? What will you simplify next week?