Use the following as the single authoritative **Codex build prompt**. It integrates the project spec (local-first Personal Knowledge Analyst) with a rock-solid Ollama plan. Follow it exactly. Where ambiguity exists, prefer determinism, privacy, and reproducibility over new features.

ROLE & MINDSET

You are a senior AI engineer tasked with building a **local, privacy-preserving Personal Knowledge Analyst**. Your job is to ship a working MVP with strong engineering discipline: deterministic outputs, strict citations, hybrid retrieval, and reliable Ollama integration. Assume **no paid APIs** and **offline-first** operation.

NORTH STAR

Deliver a grounded RAG system that ingests a small set of local sources (Markdown notes, text-layer PDFs, exported emails), indexes them with **hybrid retrieval** (BM25 + embeddings + optional reranker), and answers conversational questions with **inline citations** and a **cite-or-abstain** policy. Everything runs via **Ollama** for both chat and embeddings.

NON-NEGOTIABLE CONSTRAINTS

- **Local-only** inference: All LLM & embedding calls go to **Ollama** at a configurable base URL (default `http://localhost:11434`).
- **Strict citations & abstain**: Every answer must include source citations (doc title + line or page range). If insufficient evidence, abstain with guidance.
- **Determinism**: Fix seed/temperature for production runs; store retrieval set, prompt version hash, model names, and parameters per run to enable **replay**.
- **No OCR in v1**: Text-layer PDFs only. OCR can be added later.
- **Limited scope, perfect quality**: Only these inputs for v1— `~/KnowledgeBase/notes` (Markdown), `~/KnowledgeBase/pdfs` (text-PDF), `~/KnowledgeBase/exports/emails` (`.eml` or `.mbox`).
- **No external egress**: Do not call remote embedding or chat endpoints. Do not push documents out of the machine.

TECHNOLOGY SELECTIONS (LOCAL, CPU-FRIENDLY)

- **Chat model** (Ollama): `llama3.1:8b` (quantized) — stable baseline; later upgrade possible.
- **Embeddings** (Ollama): `nomic-embed-text` (768-d). Alternative upgrade: `mxbai-embed-large`.
- **Reranker (optional)**: `bge-reranker-base` (local cross-encoder). Flag-gated; off by default.
- **Backend**: Python 3.11+, FastAPI, Pydantic v2, Uvicorn.
- **Index & Storage**: Postgres 16 + `pgvector`; file blobs on disk. BM25 via `tantivy` (preferred) or `whoosh` fallback.
- **Frontend**: Jinja2 + Tailwind + HTMX (single-page chat with sources panel).
- **Scheduling**: APScheduler for background tasks (ingest/reindex).
- **PDF parsing**: `pdfminer.six` (text layer only).

HIGH-LEVEL ARCHITECTURE

Chat UI ↔ FastAPI ↔ Retrieval Orchestrator (BM25 + Embeddings + optional rerank) ↔ Postgres/pgvector & Tantivy index ↔ Local files. Synthesis via Ollama chat model with **JSON-only** contract. All runs stored (retrieval set, prompt hash, model params) for replay.

DATA MODEL (RELATIONAL, PGVECTOR)

Tables: `documents`, `chunks`, `qa_runs`, `qa_contexts`, `qa_answers`. - **documents**: id, path (unique), title, type (md|pdf|email|bookmark), created_at/updated_at, confidentiality_tag (default 'private'), sha256, size, meta. - **chunks**: id, document_id, ordinal, text, start_line, end_line, page_no, token_count, embedding (vector[768]). - **qa_runs**: id, question, mode (lookup|synthesize|timeline|flashcards), llm_version, prompt_version, template_hash, started_at, latency_ms, abstained. - **qa_contexts**: run_id, chunk_id, rank, score_bm25, score_embed, score_rerank, rationale. - **qa_answers**: run_id, answer_json. Indexes: ivfflat on `embedding`, and supporting indexes on foreign keys.

INGESTION PIPELINES (V1)

- **Markdown**: parse frontmatter; chunk by headings (`#`, `##`) into ~800-token chunks with 10–15% overlap; record line ranges.
- **PDF (text)**: parse with `pdfminer.six`; chunk by page/paragraph; store `page_no`.
- **Email**: parse `.eml` / `.mbox`; capture headers (From/To/Date/Subject); trim quoted text; chunk 600–800 tokens. Each chunk → add to BM25 + embeddings → insert into `chunks` with vector. Compute `sha256` on the file; skip ingest if unchanged. Provide a CLI/script to reindex specific paths.

RETRIEVAL STRATEGY (DETERMINISTIC)

1. **Intent/time normalization**: Resolve relative dates; classify mode (lookup/synthesize/timeline/flashcards) via simple rules.
2. Retrieve **BM25 top-50** and **Vector top-50**. Union + dedupe spans per document.
3. Optional reranker → top-15; else interleave BM25+Vector into top-12.
4. **Diversity cap**: max 3 chunks per document; apply **time bias** if question implies recency.
5. Build a **context pack** with snippet, citation pointer `{doc, page or Lx–Ly}`, and one-line rationale. Persist it with the run.
6. **Confidence heuristic**: normalized mix of the top-3 scores; if below threshold → abstain.

SYNTHESIS (JSON-ONLY CONTRACT)

- Enforce retrieval-only policy (“cite-or-abstain”).
- Output must be valid JSON with fields: `abstain`, `answer` (2–5 sentences), `bullets` (list), `conflicts` (list of claim+sources), `sources` (list of `{id, loc}`).
- On invalid JSON: exactly one constrained retry; otherwise return a graceful internal formatting error and store raw output for diagnostics.

FRONTEND UX (MINIMAL, POLISHED)

- Single page: left chat; right **Sources** panel. Filters: [All | Notes | PDFs | Email], date range, **Strict Mode** toggle.
- Answers show short synthesis, bullets, conflicts (if any), and a sources list with clickable citations opening a modal with highlighted snippet.
- Provide loading skeletons and empty states.

EVALS & METRICS (V1 REQUIRED)

Create a **golden set** of 20–50 real questions from the user's content. Report: - **Attribution@K** (do cited chunks actually contain the facts?), - **No-Answer accuracy** (abstain when you should), - **Key-fact match** (string/n-gram checks), - **Latency** p50/p95. Add a `make eval` command that prints a markdown table and writes `eval_report.md`.

OLLAMA INTEGRATION (WORKING PLAN)

Models to use: - Chat: `llama3.1:8b` (quantized). Enforce JSON format and deterministic settings (low temperature + fixed seed). - Embeddings: `nomic-embed-text` (768-d). Validate dimension against pgvector column.

One-knob configuration (via `.env`): - `OLLAMA_BASE_URL` (default `http://localhost:11434`), `CHAT_MODEL`, `EMBED_MODEL`, `REQUEST_TIMEOUT_S`, `STRICT_LOCAL_ONLY=true`, `RERANKER_ENABLED=false`.

Readiness gates (app startup must fail early if): 1) Daemon unreachable at `OLLAMA_BASE_URL`. 2) Required models not present locally (emit exact pull commands). 3) Embedding dimension mismatch (print expected vs actual; require reindex or model switch).

Client contracts (no code here—just behavior): - **ChatService**: Inputs (system, user, context, JSON-required flag, seed, temp, max tokens). Outputs (parsed JSON, latency, model params). Single retry on invalid JSON. - **EmbeddingService**: Inputs (list of strings), batched with shape checks; outputs vectors and diagnostics.

Diagnostics page (local only): shows Ollama base URL, models & sizes, last embedding/chat latency, context window, token usage, recent errors, and location of any raw LLM outputs stored from failures.

Validation flow (one-click): health + models + dims, embed 3 test strings, single chat turn enforcing JSON, write pass/fail report to disk.

SECURITY & PRIVACY

- All artifacts local-only. Postgres with `pgcrypto` for sensitive fields if needed; file blobs respect OS permissions.

- Default confidentiality tag `private`; retrieval excludes `secrets` unless user opts in.
- **Delete pipeline**: a single operation removes doc, chunks, vectors, and BM25 entries, and records an audit log of deletion.

RISK REGISTER & MITIGATIONS

- **Model JSON drift**: enforce JSON mode + deterministic settings + schema checker + one retry; store raw output for debugging.
- **Embedding dimension mismatch**: startup check; if mismatch detected after model switch, force reindex with explicit user confirmation.
- **Slow/unstable inference**: switch to smaller quant for chat model; cap max tokens; disable reranker; add backoff.
- **PDFs lacking text layer**: skip and surface actionable message; OCR is a stretch goal only.
- **Hallucinated citations**: use **cite-or-abstain**; penalize non-cited content; evaluate Attribution@K in CI.
- **Data duplication**: dedupe on SHA-256 at document level; chunk fingerprints to avoid duplicate spans.
- **Path & OS quirks**: normalize paths, document WSL recommendation on Windows.

DELIVERABLES (MVP COMPLETION CRITERIA)

- Running app with chat answering from local sources with strict citations and abstain behavior.
- `/health` shows all readiness checks passing.
- `/api/chat`, `/api/replay/{run_id}`, `/api/ingest/reindex`, `/api/docs/{doc_id}` implemented.
- Sources panel with clickable citations → modal highlights.
- `make index` (ingest), `make eval` (metrics), `make dev` (run), and a concise README with configuration, validation, and troubleshooting appendix.

PROJECT LAYOUT (FILES & FOLDERS)

```
pka/
  app/
    main.py
    core/{settings.py, logging.py, security.py}
    models/{db.py, schema.py}
    routers/{chat.py, search.py, docs.py, ingest.py}
    services/
      ingest/{markdown.py, pdf.py, email.py}
      index/{bm25.py, embed.py, vector.py, rerank.py}
      retrieval/{orchestrator.py, context_builder.py}
      synth/{llama_local.py, templates.py}
      evals/{scorer.py, datasets/personal_golden.yaml}
  web/templates/{base.html, chat.html, modals.html}
```

```
web/static/{css/tailwind.css, js/htmx.min.js}
infra/{docker-compose.yml, init.sql}
scripts/{reindex.py, export_run.py}
tests/{test_retrieval.py, test_citations.py}
README.md  Makefile
```

MAKE TARGETS (BEHAVIORAL SPEC)

- `setup`: create venv, install deps.
- `db-up`: start Postgres (and Ollama if using Docker compose) and apply `init.sql` (with `CREATE EXTENSION vector;`).
- `dev`: run Uvicorn.
- `index`: run reindex script over the three folders.
- `eval`: run scorer over golden set; print table and write `eval_report.md`.

ROLLOUT ORDER (SO YOU DON'T CHASE GHOSTS)

1) Implement EmbeddingService → ingest 5–10 Markdown notes only → verify vectors present. 2) Implement ChatService (JSON-only) → answer one trivial question with a forced tiny context. 3) Implement union retrieval (BM25 + Embeddings) + interleave → answer a real question with citations. 4) Add Validation flow + Evals; capture latency and JSON validity rate. 5) Add PDFs and Emails; add Strict Mode toggle and Sources modal. 6) Optional: enable reranker; profile; keep a flag to disable.

ACCEPTANCE TESTS (MANUAL)

- Ask: "What were my key takeaways from <book/note> last month?" → 2–5 sentence synthesis, bullets, and 2+ precise citations (doc + Lx–Ly or p.N–M).
- Ask a query with no support → system abstains with an actionable message.
- Click citation → modal opens with highlighted excerpt.
- Replay a run → identical output with the same seed and prompt version.
- Run `make eval` → metrics table rendered; p95 latency visible.

TROUBLESHOOTING APPENDIX (STANDARD RESPONSES)

- **Model not found**: show `ollama pull <model>` commands.
- **Daemon unreachable**: show how to start Ollama, confirm port, retry health.
- **Invalid JSON**: note one constrained retry happened; show copyable raw output and seed/prompt version.
- **Dimension mismatch**: show expected vs actual; require reindex or switch embedding model.
- **OOM/slow**: switch to smaller quant; reduce context; disable reranker; increase timeout modestly.
- **PDF skipped**: explain text layer required; suggest enabling OCR in a future version.

FIRST TASKS FOR CODEX (DO THESE NOW)

1) Create project skeleton per **Project Layout**. Add `.env` with defaults and a stubbed `/health` endpoint that returns placeholders for the three readiness checks. 2) Implement **Readiness Gates**: daemon reachability, model presence, embedding dimension check (compare against configured vector column dim). Make `/health` return pass/fail for each and fail app startup if any fail. 3) Implement **EmbeddingService** (behaviorally): batch texts, call embeddings, verify shape, and store vectors in `chunks`. Add a tiny CLI (`make index`) to ingest a handful of Markdown notes.

SECOND TASKS (AFTER GREEN HEALTH)

4) Implement BM25 index and retrieval union + interleave. Add the diversity cap and time bias. 5) Implement **ChatService** with JSON-only contract, deterministic settings (seed+temp), one retry on invalid JSON, and persistence of run/context/answer artifacts. 6) Build the minimal chat UI with Sources panel and citation modal. Then connect `/api/chat` → render. 7) Add the golden eval set + `make eval`. Log metrics into `eval_report.md`.

DEFINITION OF DONE (MVP)

- All readiness checks pass; Validation flow produces a green report.
- Ingested notes/PDFs/emails; hybrid retrieval works; answers render with citations.
- Cite-or-abstain policy enforced and visible in behavior and metrics.
- Replay produces identical results for same seed/prompt version.
- README includes configuration, validation, troubleshooting, and metrics snippet.

Implement now. Prioritize correctness (citations, abstain, replay) and reliability (Ollama health, dims, timeouts) over new features. Keep commits small and include short design notes in each PR/commit message referencing the spec sections you implemented.