Orion — Architecture & Tradeoffs

Orion Project — High-level Architecture & Tradeoffs Participants: Dev A, Dev B, Dev C Dev A: The repo is structured around document ingestion, embedding generation (dense and sparse), and a simple query interface (answer.py / query.py). We rely on Ollama locally for embedding models and ChromaDB for the dense vector index. That gives us an offline-first UX which is great for privacy and hackathon demos. Dev B: Right — offline is good for demos, but Ollama pulls and local model requirements add friction. If a contributor hasn't set up Ollama or doesn't have the right Mac setup, dense pipeline fails. An alternative is to support a remote inference fallback (OpenAI, Anthropic, or self-hosted REST) so the repo can be used in CI or cloud setups. Tradeoff: remote adds cost and privacy concerns; local adds reproducibility issues across OS. Dev C: On dense vs sparse embeddings — we have dense_embeddings.py using nomic-embed-text and sparse_embeddings.py producing TF-IDF style vectors. Dense works better for semantic queries; sparse is cheaper and explainable. But keeping both increases maintenance. I'd argue for a single canonical pipeline with a pluggable embedding interface and unit tests around expected nearest-neighbor results. Dev A: Also, ChromaDB is used for dense storage; the repo stores a pickled sparse index. Chroma is great locally, but for scaling or multi-user deployments, we'd likely migrate to a managed vector DB or Supabase + pgvector. That changes tradeoffs: reliability and replication vs cost and vendor lock-in. Dev B: There's also a UI (ui/) and Flask app. Right now the code assumes single-process local use. If we want concurrent users, we'd need to rethink state management and how embeddings/Chroma are shared across processes — e.g., run a dedicated vector store service and use REST to query it. Dev C: Summary recommendations: 1) Add a clear optional remote inference path and env var toggle (OLLAMA vs REMOTE_API). 2) Abstract embedding pipeline to support pluggable backends and tests. 3) Add a lightweight containerized dev setup (docker-compose) to remove "works on my Mac" issues.