Here’s a clear, end-to-end plan that (1) summarizes everything I’ve already built, (2) explains how each piece works, and (3) gives you a step-by-step path to a production-quality local chatbot (still privacy-friendly and free to run).

**1) What’s already built (and why it matters)**

**Core capabilities**

* Hybrid retrieval (dense FAISS + sparse BM25) with rank fusion; optional cross-encoder rerank.
* Services: swim & dryland workout generators, pace/workout analysis, injury & nutrition helpers.
* CSV↔Docs “fusion” context: athlete-specific stats stitched with retrieved doc snippets.
* CSS estimator + taper planner (evidence-aligned).
* Flexible LLM layer: run **fully local** (GPT4All/Ollama) or **no LLM** (context echo for deterministic behavior).

**Deliverables you already have**

* **Notebook (local tinkering)**: swimforge\_chatbot.ipynb (chunked, easy to edit).
* **CLI script**: swimforge\_chatbot.py (single file).
* **Local test app**: swimforge\_streamlit\_app.py (+ requirements\_streamlit.txt).
* **Seed docs** (RAG): injury, nutrition, training templates, records FAQ.
* **Seed CSVs**: athlete profiles, workouts, race results, wellness.
* **All-in-one local bundle**: swimforge\_local\_test\_bundle.zip (README + notebook + sample zips).

**Why this structure?**

* Lets you experiment fast (Notebook), chat in terminal (CLI), and demo with a simple GUI (Streamlit)—all with the same retrieval & logic stack.
* Keeps everything local (no paywalled services), and modular (so you can swap models or prompts easily).

**2) Architecture at a glance**

┌──────────────┐

│ User query │

└──────┬───────┘

│

┌─────────▼─────────┐

│ Router/Orchestr. │ (regex/intent routes: workouts, analysis,

└──────┬───────┬────┘ CSS/taper, injury, nutrition, or RAG)

│ │

┌──────────────┘ └───────────────────┐

│ │

┌────▼─────┐ ┌─────▼─────┐

│ Services │ │ RAG Core │

│ (logic) │ │ (Hybrid) │

│ swim set │ │ - BM25 │

│ dryland │ │ - FAISS │

│ analysis │ │ - RRF │

│ CSS/taper│ └─────┬─────┘

└────┬─────┘ │

│ ┌────▼──────────────┐

│ │ CSV↔Docs Fusion │

│ │ - load CSVs │

│ │ - athlete summary │

│ │ - retrieve docs │

│ └─────┬─────────────┘

│ │

│ ┌──────▼──────┐

│ │ Prompting │

│ │ (system + │

│ │ user) │

│ └──────┬──────┘

│ │

│ ┌───────▼────────┐

└──────────────────────────────────►│ LLM backend │

│ (local/none) │

└────────────────┘

**3) How each major piece works**

**Hybrid retrieval**

* Embeddings: all-MiniLM-L6-v2 (fast, free) to build a **FAISS** index.
* BM25 (rank-bm25) provides exact-term coverage for names/keywords.
* Fusion by **Reciprocal Rank Fusion (RRF)** → robust, training-free blending.
* Optional: Cross-Encoder re-rank (MS MARCO MiniLM) for tighter ordering (off by default for speed).

**CSV↔Docs Fusion**

* Detect S001 or a full name in a query → pull last 21d stats from CSVs (sessions, distance, pace/RPE, wellness).
* Retrieve top doc chunks (injury/nutrition/training templates).
* Stitch both into a single **context** for the LLM (or return context directly when LLM=none).

**Domain logic**

* **Workout generators**: parameterized by goal, stroke, volume—usable without an LLM.
* **Pace analysis**: mean/stdev/CV and rest/HR heuristics → quick feedback loop.
* **CSS/taper**: estimate CSS from trials; prescribe sets; produce taper guidelines.

**LLM layer (optional)**

* **GPT4All** or **Ollama** for fully local generation.
* Or set llm\_backend="none" for deterministic, context-only returns (useful for debugging & safety).

**Prompting**

* System + user templates that:
  + ground answers in provided context,
  + ask for specifics if context is insufficient,
  + encourage in-line “where in the doc” references.

**Safety rails**

* Injury guidance is **non-diagnostic**; the code includes referral language when pain persists or is severe.
* Nutrition is mainstream; avoids prescriptive supplement advice beyond basics.

**4) What we’ve done so far**

1. **Data scaffolding**
   * Designed four CSVs (profiles, workouts, races, wellness) + dummy rows.
   * Seeded docs/ with injury/nutrition/training/records markdown.
2. **Indexing & retrieval**
   * Implemented chunking (800 char, 120 overlap).
   * Built FAISS index + BM25 store; wired RRF fusion.
3. **Services**
   * Swim & dryland generators (parameterized).
   * Pace analysis (CV logic).
   * Injury + nutrition helpers (basic, safe defaults).
4. **Fusion**
   * Athlete detection (ID or name) → summary of recent training/wellness.
   * Combined with RAG snippets to enrich answers.
5. **CSS & taper**
   * CSS from two (or more) distance–time pairs.
   * Taper heuristics (distance-sensitive).
6. **Run targets**
   * **Notebook** for tinkering.
   * **CLI** for quick usage.
   * **Streamlit app** for a point-and-click local demo.
7. **Docs & bundles**
   * README for local testing.
   * Packaged a local test bundle zip for easy setup.

**5) Customization you can do immediately**

* **Prompts**: In the configs (CFG["prompts"]), change tone, citation rules, refusal criteria.
* **Retrievers**:
  + Toggle fusion: "rrf" vs "weighted"; adjust alpha\_dense/bm25.
  + Enable cross-encoder rerank for top-k (trade speed for quality).
* **Chunking**: Tune size/overlap for your docs (e.g., 1,000/150 if your PDFs are narrative).
* **Services**:
  + Add more goal types (e.g., “taper”, “open water”, “technique blocks”).
  + Expand pace analysis (e.g., target send-offs vs. observed).
* **LLM**:
  + Flip to gpt4all or ollama when you want fluent generation off the same context.

**6) Quality & evaluation plan (to do this as I scale the project)**

* **Retrieval metrics** (offline):
  + Build 100–200 “gold” Q/A pairs with doc IDs.
  + Compute Recall@k, MRR, nDCG@k for: BM25, FAISS, Hybrid, Hybrid+Rerank.
  + Pick k and fusion weights that maximize nDCG@10.
* **Answer review** (human panel):
  + Rubric 0–5 for factuality, citations, actionability, safety.
  + Include a coach, a physio, and a nutritionist for domain validation.
* **Unit tests** (logic):
  + CSS regression math (expected slope).
  + Pace CV thresholds.
  + Workout generator volume adds up to total\_m.
* **Telemetry (local only)**:
  + Log anonymized query type + retrieval k + refusal rate.

**8) Data governance & safety**

* Keep athlete PII minimal (IDs like S001, name mapping optional/opt-in).
* Injury & nutrition outputs should **not** be medical diagnoses; include referral prompts when red flags appear (night pain, nerve symptoms, etc.).
* For records/rules, periodically refresh docs and note “last checked” date in your knowledge base.

**9) Roadmap (next 2–6 weeks)**

**Phase 3 — Personalization & depth**

* **CSS-aware workouts**: Pull CSS per athlete from CSV (or estimate from trials) and auto-embed into workout prescriptions (send-offs/targets).
* **Training load score**: Add rolling weekly distance × RPE (e.g., monotony/strain proxy) into the structured summary.
* **Nutrition macros**: Convert post-session templates into grams/kg guidelines based on athlete body mass (if provided).

**Phase 4 — Retrieval polish**

* Add **PDF ingestion** with OCR (if needed) and figure/table extraction.
* Turn on cross-encoder re-rank only for long queries or when BM25 and FAISS disagree (heuristic toggle).
* Add **document freshness** stamps and “last-updated” per topic.

**Phase 5 — UX & packaging**

* Streamlit refinements: dropdowns for athlete ID, file upload for TT times → auto-compute CSS; sidebar switch for fusion settings.
* Export **session plans** to a printable text/markdown from the app.
* Optional: Dockerfile for fully local packaging (no cloud calls).

**Phase 6 — Pre-repo stabilization (on your signal)**

* Clean up folders into a final structure (no secrets, no sample zips).
* Add unit tests (pytest) for logic and simple retrieval smoke tests.
* Write a production README and CONTRIBUTING notes.
* Then (and only then) initialize a repo and make your first commit with a descriptive message (we already suggested good commit name/desc earlier).

Per your earlier request, I’m **not** giving you a one-click “save to repo” flow. We’ll do git formalization only when you’re ready.

**10) Concrete “do next” checklist**

1. **Run a local demo** (pick notebook / CLI / Streamlit).
2. **Swap in your real docs** (coaching notes, templates, federation PDFs), then rebuild the index.
3. **Replace CSV stubs** with real logs (even 5–10 entries help).
4. **Enable LLM** (optional): test with gpt4all or ollama to compare fluency.
5. **Tune retrieval**:
   * Try top\_k\_dense/bm25 = 8–12.
   * Compare rrf vs weighted (0.7/0.3).
   * (Optional) enable cross-encoder for top-8 rerank.
6. **Add CSS to workouts**: wire CSS pace into swim set builder for athlete-specific send-offs.
7. **Draft the evaluation set**: 50–100 realistic queries and expected source docs.
8. **Plan safety copy**: a short disclaimer injected into injury/nutrition answers.