**One-Pager: From Prototype to Production — Agentic RAG System**

| **Aspect** | **Prototype (What I Built)** | **Production-Ready (Real-World Design)** |
| --- | --- | --- |
| **RAG Architecture** | Multi-agent RAG with clustered domain agents and a meta-agent orchestrator. | Same concept retained — scaled with persistent vector stores, robust chunking, semantic caching, and token-efficient querying. |
| **System Design** | requests, BeautifulSoup, LangChain, Sentence Transformer Embedding (MiniLM), KMeans, FAISS, Gemini, pickle for cache. | Use **Airflow** or **Prefect** for ingestion flows. Use **Unstructured.io** or **Trafilatura** for parsing. Switch to **BGE embeddings** or **OpenAI/Cohere**. Index with **Weaviate** or **Qdrant** (with filters & metadata). Cache with **Redis** or **S3**. |
| **Scalability & Performance** | Agents load on-demand, clustered via KMeans, query pruning, LLM called selectively. | Use **horizontal vector index sharding**, **lazy-loading** via microservices, autoscaling API, and LLM routing optimization. Use **LangGraph** or **DSPy** for control flow. |
| **Frameworks Used** | Colab UI, LangChain chunker, FAISS, Gemini, pickle cache. | Replace Streamlit/Colab with **React + FastAPI** frontend. Use **Docker** + **Kubernetes / Cloud Run**. Full orchestration with **LangServe** or **Ray Serve**. |
| **Testing Methodology** | pytest, mock LLM calls, QA pairs, unit tests for components. | Add **integration tests**, CI/CD pipelines, use **TruLens** and **RAGAS** for automated RAG eval. Mock LLMs via local simulators or small models. |
| **Model Monitoring & Evaluation** | Logged agent decisions, LLM scoring for answer confidence. | Add **OpenTelemetry** + **Prometheus** for full-stack monitoring. Track per-agent accuracy, query latency, LLM token usage. Use **LLM feedback** to refine rankings. |
| **Error Handling** | Handled URL errors, FAISS misses, and LLM failures via try-catch. UI warnings. | Use **Sentry**, **Honeycomb**, or **Datadog** for real-time alerting. Graceful fallbacks via retry queues and async workers. |
| **User Experience** | Colab UI, upload panel, query box, confidence displayed. | Move to full-stack app with **chat widget**, feedback buttons, upload preview, streaming responses. Use **Next.js + Vercel** or embed in existing platforms. |
| **Continuous Improvement** | Plan to auto-tune cluster count, improve embedding quality, add feedback loop. | Use **auto-labeling via GPT-4** to expand QA pairs. Add **self-evaluating agents**, **LLM debates**, and **supervised fine-tuning** on user feedback. |
| **Quality Assurance** | 20+ QA queries, cross-validation of agents, meta-agent consistency checks. | Add regression suite across domains. Run **synthetic evals** on edge cases, hallucinations, and grounding. |
| **Timeline & Estimation** | ~2.5–3 weeks for full system with prototype stack. | ~4–6 weeks with production stack, CI/CD, frontend, and full logging/monitoring integration. Additional 2–4 weeks for hardening and rollout. |
| **Architecture Cost** | $20–45/month: Gemini Flash + local cache + FAISS | $100–500/month depending on scale. Replace FAISS with managed vector DB (Weaviate, Pinecone). Use LLM quotas, autoscaling LLM calls, and global CDN for performance. |

|  |  |  |
| --- | --- | --- |
| **Metric** | **Prototype Estimate** | **Scalable Production Setup** |
| Latency per query | ~500–1000 ms | 200–800 ms with async + streaming |
| Throughput | ~1–2 QPS (local) | 20–100+ QPS with horizontal scaling |
| Cost per query | ~$0.002–$0.005 | Controlled via token limits & caching |
| Indexing time | ~30–60 sec per 100 docs | Parallelized ingestion |
| Evaluation | Manual + confidence scoring | RAGAS, synthetic QA evals |
| Vector DB | FAISS (in-memory) | Weaviate / Pinecone / Qdrant |

**For 1 million users, assuming 5% daily actives and 5 queries per user, we’d expect ~250,000 read requests per day and ~5,000 document upload requests. On the storage side, if each user uploads just two documents, we’d need roughly 9 TB of storage to hold raw documents, parsed text, vector embeddings, and metadata. This helps us plan storage tiers and auto-scaling — using S3 for raw input, Weaviate or Pinecone for vector indexes, and Redis or Postgres for fast metadata access.**

|  |  |
| --- | --- |
| **Parameter** | **Estimate** |
| Daily Active Users (5%) | 50,000 users |
| Daily Read Queries (/ask) | 250,000 queries |
| Daily Uploads (/upload-docs) | ~5,000 documents |
| Avg. Docs/User (lifetime) | 2 docs |
| Total Docs | 2 million documents |
| Storage – Raw (1 MB/doc) | ~2 TB |
| Storage – Parsed (0.5 MB/doc) | ~1 TB |
| Storage – Embeddings (10 chunks × 200 KB) | ~4 TB |
| Storage – Metadata/Index | ~2 TB |
| Total Storage Estimate | ~9 TB |
| LLM Cost per Query | ~$0.003 avg × 250K → ~$750/day = ~$22.5K/month |
| Infra Cost (Vector DB, Storage) | ~$500–1500/month (Weaviate, S3, Redis, etc.) |
| Total Monthly Ops Cost | ~$2,000–$3,000 base + $20–25K LLM usage = ~$25–28K/mo |

**Suggested Pricing Model: Per 1,000 Queries**

| **Tier** | **Cost to You (Est.)** | **Suggested Price** | **Profit Margin** |
| --- | --- | --- | --- |
| 1,000 queries | ~$3–5 | $10–15 | ~2×–3× markup |
| 10,000 queries | ~$30–50 | $100–150 | ~3× |
| 100,000 queries | ~$300–500 | $800–1200 | ~2.5× |
| 1M queries | ~$3,000–5,000 | $8,000–12,000 | ~2.5×–3× |

**Alternative: Per User Monthly License**

| **Users** | **Est. Monthly Cost** | **Suggested Charge** | **Profit Margin** |
| --- | --- | --- | --- |
| **1,000 users** | **~$700** | **$2,000** | **~65%** |
| **10,000 users** | **~$7,000** | **$20,000** | **~65%** |
| **50,000 users** | **~$17,000–22,000** | **$50,000+** | **~55–60%** |