RAG People Finder - Design Note & Deployment Guide

Dataset Creation

Created a 700-row synthetic dataset using domain-specific templates instead of random text generation. Each founder profile includes coherent combinations of keywords, startup ideas, and professional backgrounds across 15 domains (AI, healthtech, fintech, etc.). Used Faker for names/emails and custom templates like "developing AI-powered solutions for automated business process optimization" rather than meaningless random sentences. Company names are domain-appropriate (NeuralLabs for AI, CareTech for healthtech) ensuring semantic consistency for RAG retrieval.

Technical Stack

- **Frontend**: Streamlit for rapid prototyping and deployment
- **Embeddings**: SentenceTransformer (all-MiniLM-L6-v2) for 384-dimensional semantic vectors
- **Search**: Hybrid approach combining cosine similarity + BM25 keyword matching
- **Location**: Dynamic fuzzy matching with abbreviation handling (blr→Bangalore)
- **Storage**: CSV + NumPy arrays for embeddings, optimized for <1GB memory footprint
- **Dependencies**: pandas, scikit-learn, rank-bm25, fuzzywuzzy

Architecture & Indexing Pipeline

Two-stage filtering: (1) Mandatory location filtering using fuzzy string matching and semantic similarity, (2) Advanced RAG on location-filtered subset using fusion retrieval combining 5 strategies (semantic search, query expansion, HyDE, BM25, field-specific search) with Reciprocal Rank Fusion scoring. Embeddings pre-computed offline: individual field embeddings + combined embeddings for idea+about+keywords. No real-time embedding generation during search for performance.

Local Setup

```
git clone [repo-url]
cd rag-people-finder
pip install -r requirements.txt
python generate_dataset.py  # Creates synthetic data
python indexing.py          # Generates embeddings
streamlit run app.py        # Launches on localhost:8501
```

Hosting Strategy

Deploy on **Streamlit Cloud** (streamlit.io) for simplicity, connecting directly to GitHub repo for automatic deployments. For production scale, containerize with Docker and deploy on **Google Cloud Run** or **AWS ECS Fargate** for serverless scaling. Cloud Run handles HTTPS automatically and scales to zero when unused, cost-effective for demo workloads. Store embeddings in Cloud Storage/S3, mount as volumes during container startup. Alternative: Railway.app or Render.com for simpler deployment without container management.

Authentication & Security

Implement **Streamlit-Authenticator** for simple username/password auth stored in encrypted YAML config, or integrate **Auth0** for OAuth (Google/GitHub login) using streamlit-oauth library. Auth logic lives within Streamlit app rather than reverse proxy for simplicity. Store API keys and secrets in environment variables loaded from .streamlit/secrets.toml locally or cloud provider's secret management (Google Secret Manager, AWS Systems Manager). Enable HTTPS through cloud provider's load balancer with automatic SSL certificates.

Production Operations

For safe re-indexing: (1) Generate new embeddings in separate directory, (2) Atomic swap using symlinks or container image updates, (3) Health checks to ensure embedding dimensions match before serving traffic. Use Cloud Storage versioning for rollback capability. Environment variables: EMBEDDING_PATH, MODEL_NAME, DEBUG_MODE. Monitor with cloud provider metrics (request latency, memory usage). Consider read replicas for high availability and implement graceful degradation if embedding service fails.

Edge Cases Handled

- **Location variations**: "blr"→"Bangalore", typos like "bangalor"
- **Empty queries**: Requires location specification with helpful error messages
- **No matches**: Provides alternative city suggestions from dataset
- **Missing embeddings**: Graceful fallback to keyword search only
- **Large result sets**: Pagination and relevance scoring to surface best matches
- **Ambiguous queries**: Query expansion and multiple retrieval strategies
- **Memory constraints**: Lazy loading of embeddings, efficient NumPy operations

---

Demo URL: Enhanced RAG People Finder · Streamlit
GitHub: eshaangolatkar/RAG-Chat