Recommended Approach for POC → FAISS or ChromaDB

- **Use FAISS** if you want **fast, local retrieval** (best for MVP).
- **Use ChromaDB** if you need **metadata filtering** (e.g., search by `"price < ₹1000"`).
- **If scaling beyond 1M entries**, move to **Pinecone or Weaviate**.

Comparison of Vector Databases:

Vector DB	Best For	Pros	Cons
FAISS (Facebook AI Similarity Search)	Local, fast search, small to medium datasets	Super fast, runs locally, efficient	No built-in filtering
ChromaDB	Simple, easy to use, metadata filtering	Open-source, native LangChain support	Slightly slower than FAISS
Pinecone	Scalable, cloud-based search	Handles large-scale data well	Paid after free tier
Weaviate	Hybrid search (text + vectors)	Strong metadata filtering	More setup require

Feature	FAISS	ChromaDB
Speed	✅ Very Fast	🚀 Fast, but slightly slower than FAISS
Scalability	✅ Handles millions of records	✅ Scales well, but needs metadata indexing
Metadata Filtering	❌ No (must filter after retrieval)	✅ Yes (can filter before retrieval)
Ease of Use	✅ Simple setup	✅ Simple + LangChain support
Best Use Case	Pure similarity search	Hybrid search (vectors + structured filters)

Faiss

Chunking is important for breaking down large text data into meaningful pieces before converting them into vector embeddings. Your choice depends on the **type of content and retrieval use case**.

📌 *Basic Chunking Strategies:*

Chunking Type	Best For	Example
Fixed-Length	General text search, structured text	512 tokens per chunk
Semantic-Based	Long, complex documents where sentences are connected	Splitting at paragraph level
Recursive Chunking	When text is hierarchical (e.g., heading → subheading → content)	First by section, then smaller chunks

How to choose?

- **Short FAQs or structured text?** → Use **Fixed-Length Chunking**.
- **Long articles with meaningful sections?** → Use **Semantic-Based Chunking**.
- **Hierarchical documents (like contracts or research papers)?** → Use **Recursive Chunking**.

1. Indexing Methods (Storage & Retrieval)

Index Type	Description	Best Use Case
IndexFlat	Brute-force search (stores all vectors)	Small datasets, 100% accuracy
IndexIVFFlat	Clusters vectors into groups before searching	Large datasets, faster than Flat
IndexIVFPQ	Compressed storage with PQ (Product Quantization)	Large-scale, memory-efficient
IndexHNSW	Uses Hierarchical Navigable Small Worlds for fast search	Faster than IVF for real-time queries

IndexLSH	Uses Locality-Sensitive Hashing	Approximate nearest neighbor search

Search Distance Metrics

- **L2 (Euclidean Distance)** → Measures absolute differences.
- **Inner Product** → Best for similarity-based ranking.
- **Cosine Similarity** → Normalized similarity metric, good for NLP.

GPU Acceleration

- Can use **GPUs to speed up search** on large datasets.
- Supports **multi-GPU parallel processing**.

◇ *4. Hybrid Search (FAISS + Metadata)*

Since FAISS doesn't support metadata filtering, you can:

- **Store metadata in a separate database (e.g., PostgreSQL, MongoDB)**.
- **Filter results after FAISS retrieves similar embeddings**.

Resource :

https://medium.com/@vivekuni7/choosing-the-right-vector-database-for-production-a-comprehensive-guide-e8f5bed109e7

https://medium.com/@punya8147_26846/comparing-faiss-and-chromadb-1428d8886506