Alexandrian: Full System PRD Overview for Codex

This document provides a comprehensive, implementation-ready specification of the Alexandrian RAG framework. It is designed to be used by a code-generation agent like Codex to scaffold, implement, and test the entire system.

🎯 Mission

Alexandrian is a self-contained, developer-first "RAG-in-a-box" that installs, embeds, clusters, and queries source projects using only local resources. It operates via an elegant CLI and API, supports zero cloud dependence, and is optimized for integration with tools like Cursor and VS Code.

⚙️ Core Components

1. **Embedding Engine**

* Uses `bge-small-en` by default via `sentence-transformers`
* Exposes `embed_text()` and `embed_text_batch()`
* CLI-accessible: `alexandrian embed --text "..."`
* Output: 1536-dim float array per input
* Configurable via `alexandrian.config.json`

2. **Vector Store**

* Uses Qdrant (Dockerized)
* Supports:

  * `upsert(vector, metadata)`
  * `search(query_vector)`
  * `update_payload(id, cluster_id)`
  * `get_all_vectors()`

3. **Clustering Engine**

* Uses HDBSCAN to tag vectors with `cluster_id`
* Optional UMAP for visualization
* Exposed via CLI and API (`/recluster`)

4. **Prompt Builder**

* Selects top-N similar chunks from Qdrant
* Inserts into prompt template
* Trims to fit token budget

5. **LLM Engine**

* Local Ollama model (default: `mistral`)
* Fast API endpoint: `http://localhost:11434`
* Handles `respond(prompt) -> str`

6. **Chunker**

* Code-aware splitter
* Splits by function/class/markdown heading
* Returns chunk metadata: file, line, cluster

7. **CLI Interface** (`alexandrian`)

Commands:

```bash
alexandrian ask "..."
alexandrian import ./src
alexandrian recluster
alexandrian embed --text "..."
alexandrian purge
alexandrian status
```

8. **FastAPI Backend**

Routes:

```http
POST /ask
POST /import/file
POST /import/folder
POST /recluster
GET  /status
GET  /search
```

9. **Config Layer**

* JSON file: `alexandrian.config.json`
* Stores:

  * model paths
  * ports
  * embedding dimensions
  * clustering params

10. **Installer**

* Command: `npx alexandrian create`
* Does the following:

  * Installs: Docker, Ollama, dependencies
  * Pulls models: Mistral + BGE
  * Bootstraps `alexandrian.config.json`
  * Starts all services

✅ Defaults

Component	Default
LLM	`mistral` via Ollama
Embedding	`bge-small-en`
Clusterer	`hdbscan` + UMAP
Framework	FastAPI + uvicorn
Vector DB	Qdrant

🔧 Auto-Installed Dependencies

```txt
fastapi
uvicorn
qdrant-client
sentence-transformers
hdbscan
numpy
scikit-learn
httpx
```

ollama
docker

🧪 Postinstall Test

After running `alexandrian create`, verify with:

```bash
alexandrian status
```

Returns:

```json
{
  "qdrant": "healthy",
  "ollama": "running",
  "embedding_model": "bge-small-en",
  "cluster_count": 12,
  "total_chunks": 143,
  "last_ingest": "2025-07-26T14:22:00Z"
}
```

🔍 Optional Features (Stretch Goals)

* UMAP cluster explorer UI
* GitHub listener for re-index
* VS Code plugin
* Web UI
* Token/cluster visual analytics
* Memory cache for frequent queries

📦 Ready for Implementation

Each component is self-contained, testable, and regenerable. Codex should treat each module as a standalone function and generate all endpoints, CLI commands, test cases, and startup scripts.

Modules may be developed in isolation or as layers within a monorepo.
