FileNest Framework

A Multi-Modal Intelligent Search Engine over Decentralized Data

Summer RAID ’25

Project Proposal

Submission Date	June 5, 2025
Program	Summer RAID 2025
Project Domain	Multimodal AI and Semantic Retrieval + Peer-to-Peer Networks
Mentor	Lakshya Jain	Aradhya Mahajan	Laksh Mendpara
Institution	IIT Jodhpur

1 Project Overview

FileNest Framework: A Multi-Modal Intelligent Search Engine over Decentralized Data

1.1 Executive Summary

FileNest Framework represents a paradigm shift in information retrieval and content discovery, leveraging cutting-edge artificial intelligence and distributed computing technologies. The platform addresses the growing need for intelligent, scalable, and semantic search capabilities across diverse content types including text documents, research papers, images, and multimedia content. By implementing a peer-to-peer distributed architecture with advanced AI-powered understanding, FileNest eliminates traditional bottlenecks while providing unprecedented search accuracy and performance.

2 System Overview (Work in Progress)

At a high level, FileNest consists of a hierarchical distributed tagging system that combines intelligent embedding generation with peer-to-peer routing mechanisms. The system operates through four distinct phases that work together to create a scalable, intelligent, and decentralized search infrastructure.

2.1 Architecture Components

2.1.1 Bootstrap Phase (Depth 1 Tagging Vector Creation)

- Generate a set of sample embeddings from representative files across the network
- Cluster or otherwise select 100 initial "Depth 1 Tagging Vectors" (D1TVs).
- Distribute those 100 D1TVs (and their responsibility assignments) to all peers.

2.1.2 Local Indexing Phase (per peer, for each file)

- A peer generates an embedding (n-dimensional float vector) for a file in its shared directory.
- It finds the most similar Depth 1 tagging vector (via cosine similarity).
- It "routes" the file metadata (peerID, fileURI, embedding) to the responsible Depth 1 peer, incrementing that D1TV's count and retraining the D1TV vector asynchronously.
- That peer (Depth 1) repeats the same logic at Depth 2, and so on, until Depth 4. At Depth 4, the file metadata is finally stored at the leaf peer.

2.1.3 Continuous Training (Tagging Vector Updates)

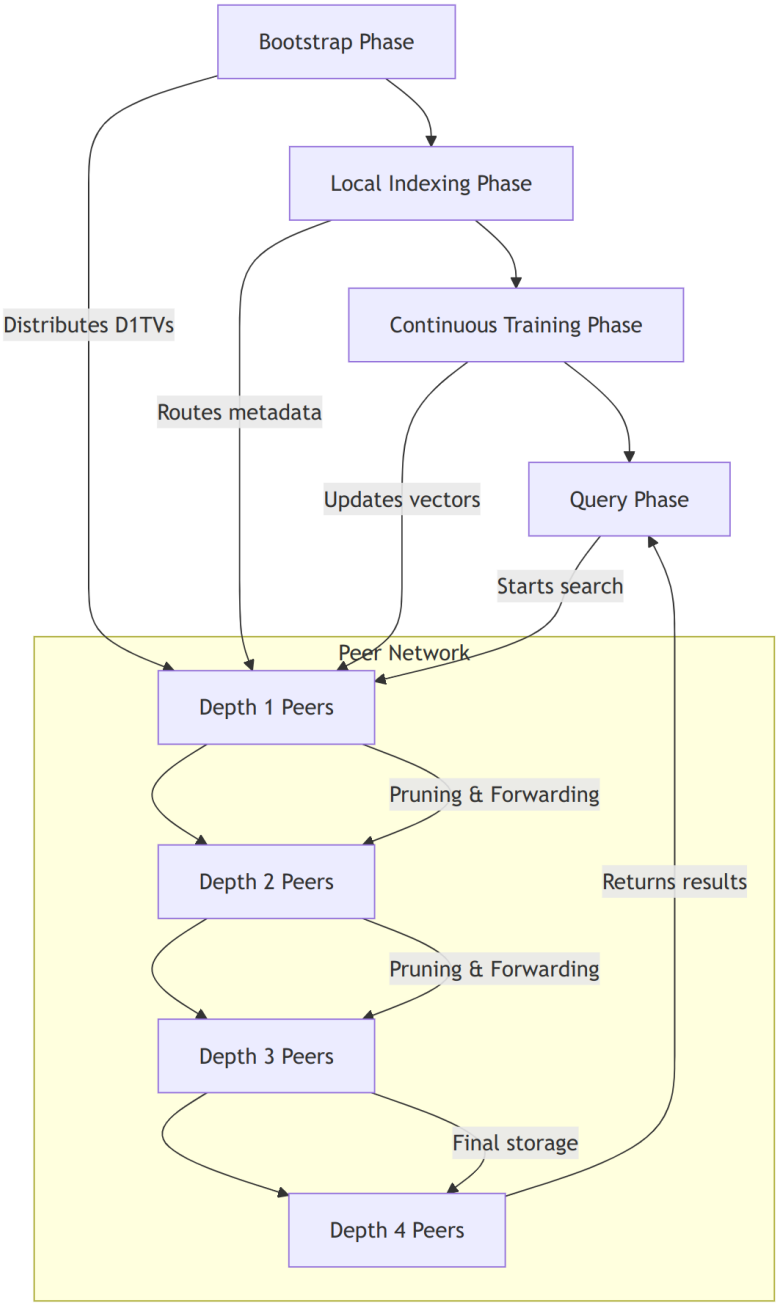
- Whenever a new embedding is routed to an existing tagging vector (similarity > threshold), the tagging vector's centroid is updated via a weighted mean function.
- If no existing tagging vector at a given depth is similar enough (similarity < threshold), a new vector is created at that depth. A peer is assigned via DHT to own that new vector.

2.1.4 Query Phase (from a client)

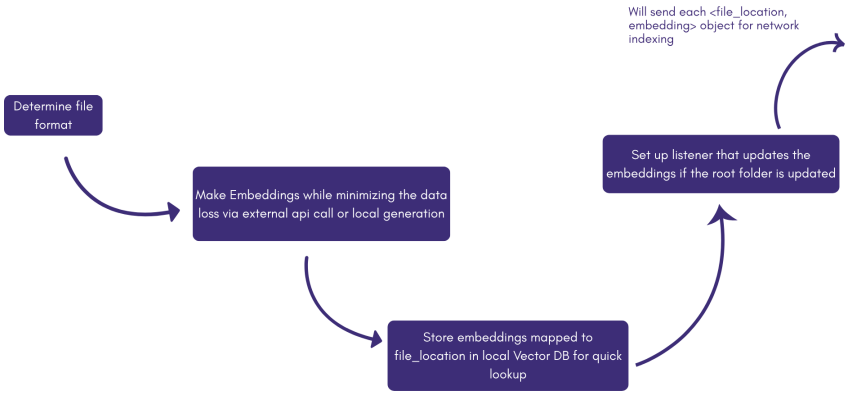
- The client computes an embedding for its query (text, image, etc.) using the same ML model.
- The client also specifies per-depth similarity thresholds.

- The embedding is sent to all Depth 1 peers. Each Depth 1 peer computes similarity vs. its tagging vectors; if the maximum similarity greater than the client's Depth 1 threshold, that branch "survives," otherwise it is pruned.
- The surviving Depth 1 peers forward the query to Depth 2 peers, applying the Depth 2 threshold, and so on down to Depth 4.
- Each Depth 4 (leaf) peer that is reached returns file matches (fileURI, originPeer, similarity). The client aggregates results.

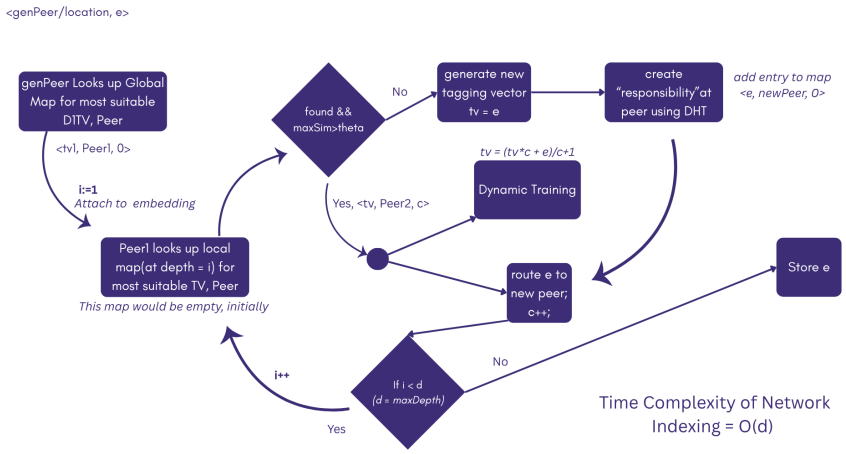
2.2 System Architecture Diagram



2.2.1 Local Indexing



2.2.2 Network Indexing



3 Objective

Current search and indexing solutions face significant limitations in handling diverse content types, providing semantic understanding, and scaling efficiently across distributed environments. Existing platforms like Elasticsearch, Apache Solr, and Algolia suffer from centralized architecture bottlenecks, limited multi-modal support, and high infrastructure costs that make them unsuitable for modern AI-powered applications.

In an age where most information access is mediated by centralized entities—search engines, cloud storage providers, content hosting platforms—we've grown used to convenience at the cost of control. These systems, while efficient, come with a host of problems:

3.1 Problems Addressed

- **Data centralization** leads to privacy risks, data mining, and platform censorship.
- **Limited access** to information that goes against dominant narratives or is locked behind content policies
- **Insufficient Academic Integration**: Lack of specialized algorithms for research paper understanding, citation analysis, and scientific content discovery
- **Fragile hosting** models—where files and knowledge vanish when servers shut down or accounts are banned.

- **Growing mistrust** in the algorithmic filters that dictate what we see.

FileNest aims to flip that model. It provides an intelligent search engine that runs across a decentralized, peer-based network. It combines the ease of intelligent discovery—like Google Drive Search or YouTube recommendations—with the resilience and freedom of P2P systems.

3.2 Real-World Impact

This opens up exciting possibilities for niche yet critical use-cases:

- **Academic collaboration**: Labs or researchers can share files and papers directly, searchable across institutions without cloud lock-in.
- **Censorship-resistant content discovery**: Useful for journalists, archivists, or individuals seeking sensitive or restricted knowledge.
- **Decentralized knowledge commons**: Like a smarter, distributed version of torrenting — but semantically searchable.
- **Self-hosted file search across devices or teams**: Enabling secure internal discovery without exposing to external servers.

The core idea: Even though data is decentralized and scattered, FileNest makes discovering it as simple and powerful as searching Google — but without giving up ownership, privacy, or freedom.

4 What We Are Trying to Achieve

4.1 Primary Objectives

1. **Performance Optimization**: Achieve 400-500% improvement in indexing throughput and search response times compared to current implementation of local-first intelligent search
2. **Multi-Modal Intelligence**: Develop advanced AI-powered content understanding for text, images, videos, PDFs and other file formats
3. **Decentralized Architecture**: Build a peer-to-peer network system that creates fault tolerance and reduces central dependency
4. **Academic Specialization**: Create sophisticated research paper analysis capabilities with citation networks and semantic understanding
5. **Scalability Enhancement**: Enable enterprise-grade deployments supporting thousands of concurrent users

4.2 Technical Goals

- Reduce search latency from 800-1200ms to under 100ms
- Increase indexing throughput from 2-3 files/sec to 15-20 files/sec
- Support multi-modal content types with cross-modal search capabilities
- Implement consensus algorithms for data integrity
- Achieve storage efficiency with less than 20% overhead

5 Expected Outcome

5.1 Primary Deliverables

1. **Multi-Modal Search Engine**: Production-ready system supporting text, image, video, and PDF content with semantic understanding.
2. **Research Paper Intelligence**: Specialized module for academic content analysis, including citation extraction and relationship mapping.
3. **Distributed Computing Platform**: Peer-to-peer network architecture with consensus mechanisms and fault tolerance for high availability.
4. **Natural Language Interface**: AI-powered query processing system enabling complex natural language searches with contextual understanding.
5. **Performance Optimization Suite**: Advanced algorithms and caching mechanisms delivering sub-100ms search responses for optimal user experience.
6. **DevOps Integration Framework**: Robust CI/CD pipelines, automated testing suites, and monitoring tools ensuring code quality, system reliability, and faster deployment cycles.

5.2 Measurable Outcomes

Metric	Current State	Expected Outcome
Indexing Speed	1 file/second	15-20 files/second
Search Latency	800-1200 milliseconds	Under 100 milliseconds
Content Types	All	All
Concurrent Users	Single user	1000+ users
Storage Overhead	85%	Under 20%
API Dependencies	100% external	50% self-reliant

5.3 Innovation Impact

The project will establish new benchmarks for distributed search architecture while demonstrating the feasibility of AI-powered multi-modal content understanding in production environments.

6 Tech Stack

6.1 Backend & Network Infrastructure

- **Core Language**: Golang for local-first application logic and integration of AI/ML components
- **Network and Communication**: go-libp2p-kad-dht (or similar custom implementation) for discovery and network routing
- **Database Layer**: SQLite/PostgreSQL for metadata storage, vector databases (Milvus/FAISS) for embeddings
- **Caching System**: Redis (Modularized) for high-performance caching and session management

6.2 AI/ML Components

- **NLP Frameworks**: Hugging Face Transformers library with models like BERT, RoBERTa, T5 for semantic understanding and query processing
- **Embedding Generation**: Sentence-Transformers (e.g., all-MiniLM, paraphrase models) for efficient and compact semantic vector representations
- **Embedding Optimization**: Techniques for dimensionality reduction, quantization, and indexing using FAISS and Milvus
- **Computer Vision**: OpenCV and PIL for preprocessing; CLIP for cross-modal (image-text) embedding and similarity
- **Multimodal Learning**: Integration of image, text, and video embeddings for unified search and retrieval

6.3 Client-side UI and Frontend

- **Executable Binaries**: Golang Binary Executables with Cobra CLI
- **Frontend Framework**: React.js for user interface development

7 Team Composition and Expertise Requirements

7.1 Optimal Team Structure

Recommended Team Size: 7 enthusiastic and curious members, with interests across AI/ML, backend development, networking, DevOps, and frontend/UI. While roles are defined for clarity, the team is encouraged to learn across domains and collaborate freely.

7.2 Core Team Roles and Responsibilities

7.2.1 AI/ML Research Engineers

- **Primary Responsibilities**:
  - Design and optimize semantic embedding pipelines for heterogeneous data types including text, images, and videos
  - Develop advanced natural language processing modules for semantic query understanding, document ranking, and content summarization
  - Architect and implement AI agents capable of multimodal content analysis, enabling unified search across diverse file formats
  - Integrate and fine-tune vector-based similarity search mechanisms to support fast and context-aware retrieval in decentralized environments
- **Learning Opportunities**:
  - Direct involvement in the end-to-end development of intelligent retrieval systems operating over decentralized, peer-to-peer architectures
  - In-depth exposure to state-of-the-art multimodal learning techniques, including cross-modal representation alignment and scalable vector-based retrieval frameworks
  - Active contribution to a research-intensive, open-source initiative advancing the integration of natural language processing, computer vision, and distributed systems for next-generation AI applications

7.2.2 Backend and Networking Developers

- **Primary Responsibilities**:
  - Build core backend logic integrating local search with a distributed P2P protocol
  - Help design the federated tag-based query routing system
  - Implement APIs, file retrieval, and metadata serving
- **Learning Opportunities**:
  - Work on a scalable, intelligent system using real backend patterns
  - Understand decentralization, content routing, and federated search design

7.2.3 UI/UX and Frontend Developer

- **Primary Responsibilities**:
  - Design and implement user-facing interfaces — CLI tools and/or web-based clients
  - Ensure the search experience is intuitive, usable, and accessible
  - Integrate metadata previews, search bar UX, file downloads, and more
- **Learning Opportunities**:
  - Understand the frontend side of an intelligent system
  - Create usable tools over decentralized protocols

7.3 Note on Collaboration and Learning

While the roles above are defined to organize efforts and expertise areas, every team member is encouraged to:

- Learn from adjacent domains
- Participate in cross-functional problem solving
- Contribute wherever your curiosity takes you
- **Spoon-feeding is not expected**

This is an open-source, learning-driven project — the goal is growth, exposure, and building something awesome together.

8 Milestones and Timeline

8.1 Overview

The development of FileNest will be carried out in two parallel tracks:

1. **Track 1 – Local Intelligent Search Engine:** Building an agentic AI-based system to index and search files locally using embeddings.
2. **Track 2 – Nesting Protocol and Network Layer:** Developing a federated, tag-based P2P protocol for decentralized querying and retrieval.

Both tracks will progress simultaneously with active collaboration and even switching among team members, coordinated by mentors to ensure modularity, interoperability, and a strong learning-driven development experience.

8.2 Timeline Breakdown (12 Weeks)

- **Week 1: Learning and Planning Phase**
  - All mentees onboarded with basics of ML, embeddings, P2P systems, backend architecture, and DevOps
  - Mentors define and finalize architecture, module plan, and target technologies
  - Establish internal documentation and shared knowledge base
- **Week 2: Environment Setup and Protocol Design**
  - Set up dev environments, tools, and repos
  - Define modular code structure and interface formats for message passing, metadata representation, and file embedding schemas
  - All mentees onboarded with task-specific knowledge, programming language and concepts
  - Establish coding practices, documentation guidelines, and GitHub workflows
  - Begin scaffolding core components of both tracks
- **Week 3-6: Parallel Module Development**
  - **Track 1**: Implement agentic AI-based local file analysis and indexing (text, image, video)
  - **Track 2**: Begin implementation of basic P2P layer, including node communication and tag-based routing hierarchy
  - Conduct daily or near-daily syncs to maintain alignment across tracks
  - Each module to be tested and documented upon completion
- **Week 7-8: System Testing and Integration Prep**
  - Test local search engine independently for accuracy, speed, and usability
  - Simulate the network on a single machine using multiple processes to verify nesting and routing logic
  - Begin testing over LAN with multiple machines for OS and hardware compatibility
- **Week 9: System Integration and LAN Testing**
  - Integrate both tracks into a unified system
  - Ensure seamless communication between local search and network query layer
  - Use mDNS for peer discovery within LAN
  - Conduct preliminary full-system LAN tests
- **Week 10: Optimization and Load Testing**
  - Identify potential optimizations across both AI and network components
  - Conduct stress and load testing to evaluate system limits
  - Compare results with expected benchmarks and document performance
- **Week 11-12: Deployment and Finalization**
  - Implement advanced network abstractions to support deployment beyond LAN (e.g., NAT traversal, optional relays)
  - Deploy working prototype over cloud or public testbed
  - Set up CI/CD pipelines and finalize project documentation
  - Prepare project showcase, tutorials, and demo materials

9 Learning Resources

- Deep Learning Fundamentals by Sebastian Raschka
- Sequence Models (Deep Learning Specialization) by Andrew Ng
- How Peer to Peer (P2P) Network works
- But how does Bitcoin actually work? | 3Blue1Brown

Submitted to **Summer RAID 2025**
Realm of Artificial Intelligence and Data