1. Objective

To build a semantic search application that allows users to ask natural language queries and retrieve relevant content from a multimodal knowledge base consisting of text, images, audio, and video. The app will support domain-specific queries related to:

- NAGFORM
- NAGSIM.2D
- NAGSIM.3D

The system will use LLaMA 3.2 Vision (preferred) or equivalent open-source models to enable semantic understanding and multimodal indexing/searching.

2. Scope of the Project

Develop a multimodal semantic search application that uses LLaMA 3.2 (preferably) Vision LLM to search across a domain-specific knowledge base consisting of:

Inputs to the System

Source Type	Formats Accepted	Processing Action
Documents	PDF, Word, CSV, TXT	OCR (if needed), chunking, text/image separation, embedding
Images/Diagrams	PNG, JPG, SVG, etc.	Captioning (BLIP/LLaVA), embedding, visual indexing
Audio	MP3, WAV, etc.	Transcription (Whisper), timestamping, chunking

Video	MP4, AVI, WebM	Audio extraction + keyframe analysis + transcription

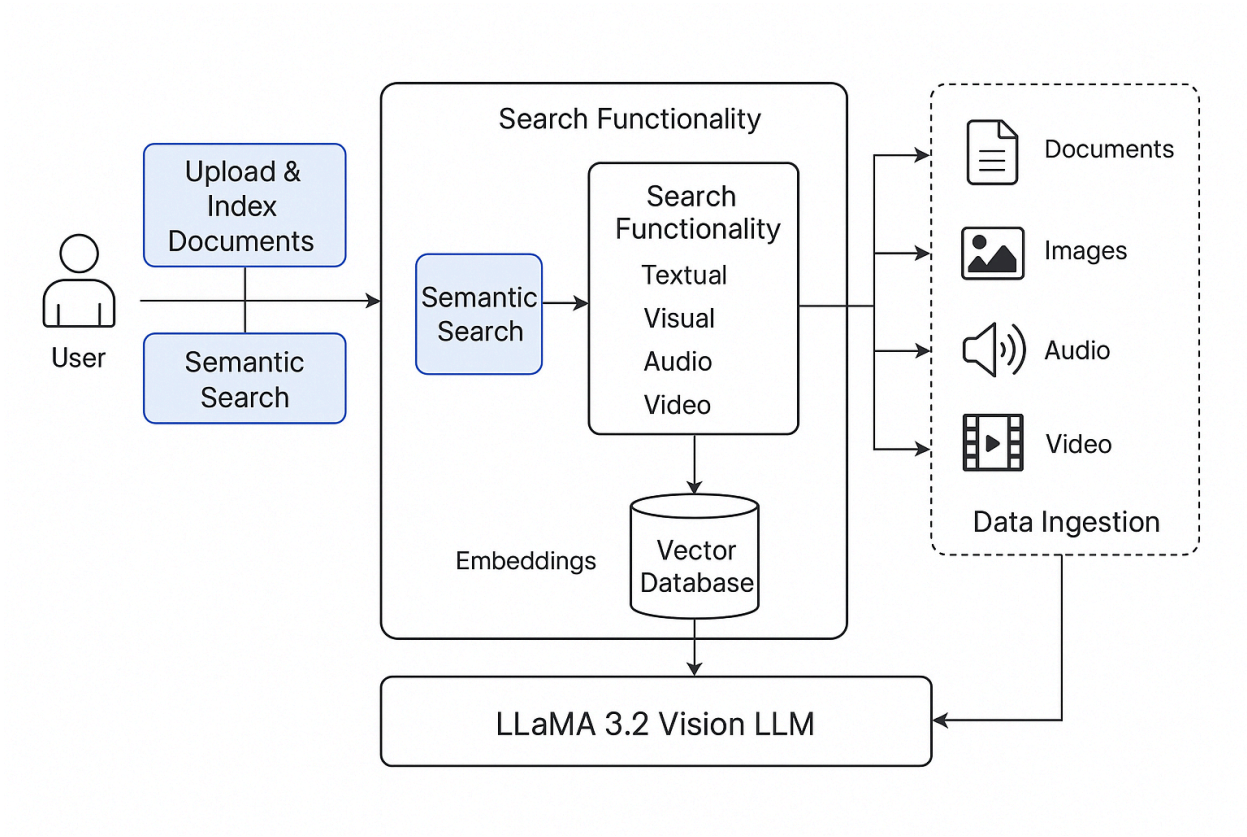
Outputs of the System

Output Type	**Description**
Semantic Results	List of ranked matches from multiple modalities
Text Snippets	Paragraphs/sections from PDFs, manuals, or training docs
Image Previews	Captioned or related image excerpts (e.g., part geometry)
Timestamped Clips	Video/audio segments with playback starting at relevant time
Source Metadata	File name, section title, page number, timestamp, etc.

2.1 Core Features & Capabilities

1. **Multimodal Search -** Accept **natural language queries** from the user.
2. Retrieve **semantically relevant segments** from: **Textual** content, **Visual** content, **Audio**.

3. System Architecture



3.1 Ingestion Pipeline

Task	Details
File upload/import	Upload via UI or auto-sync with local file system
Preprocessing	File conversion, OCR , parsing
Chunking	Split long text/audio/video into semantic chunks
Caption generation	For images
Transcription	Audio and video to text
Embedding generation	Text: LLaMA3 or SentenceTransformer, Image: CLIP, Audio: Transcript
Metadata tagging	Attach doc_id, page number, timestamps, source type, etc.

3.2 Document Chunking & Annotation

- Smart **chunking** of documents (page, paragraph, or visual section)
- For multimedia:
    - Extract **keyframes** and **transcriptions** (for voice/video)
    - Annotate **timestamps**, **speakers**, and **visual descriptions**

3.3 Embedding

Embedding-Based Retrieval - Extract embeddings using modality-specific encoders

- Textual data: Sentence-transformer or similar LLaMA-compatible text encoder
- Visuals: Vision transformer within LLaMA 3.2
- Audio/Video: Use whisper-compatible audio embedding or convert to text with timestamps, then embed

Store these embeddings in a vector database (e.g., FAISS, Weaviate, Pinecone)

3.4. Indexing

- Indexed with full metadata for linking to the source files in **MySQL**
- For every chunk extracted from your data (PDF, image caption, audio transcript, etc.), you create a vector and attach metadata.
- **Example of Metadata to Store with Each Chunk**

Field	Purpose
doc_id	The unique ID of the document in your MySQL DB
chunk_id	Internal ID or sequence number (e.g. 1, 2, 3...)
source_type	e.g. "pdf", "image", "audio", "video"
page_number	For PDFs
timestamp_start	For audio/video (in seconds)
filename	Name of original file
title	Document title or section header

url (optional)	If the document is web-based or externally stored

- This metadata is stored **alongside the embedding vector** in the vector DB
- The user enters a query.
- App gets the most relevant **vector chunks** using semantic search.
- It reads the **metadata** for each matching chunk.
- Using the doc_id, your app can:
  Fetch full document details from **MySQL**
  Highlight the matching section (chunk)
  Jump to the correct **timestamp** (in case of audio/video)

3.5 Semantic Matching

- Retrieve results using approximate nearest neighbor (ANN) search or distance metrics (cosine similarity).
- Return a ranked list of segments (text/images/snippets from audio/video)
- The search returns results (data objects or pointers to them) based on similarity scores.
- Optionally, a large language model (LLM) or other model can be used to rerank the results based on context and relevance.

4. Query Workflow

Step	**Task Description**
User input	User enters a natural language query (typed or spoken)
Query embedding	Use LLaMA 3.2 Vision (or alt..) to create query embedding
Vector search	Find the most similar chunks from the vector DB (semantic search)
Rank + Filter	Rank by similarity score, optionally filter by modality/file type
Link to source	Use metadata (`doc_id`, page, timestamp) to retrieve the full result from MySQL
Display	Show result snippet + source + preview (text, image, audio, video)

5. System Components and Technical Stack

Component	Technology Stack
Frontend UI	React / Streamlit / Next.js
Backend API	FastAPI / Flask
Vector DB	FAISS / Qdrant / Weaviate
Document DB	MySQL / Postgres
File Storage	Local / S3-compatible blob storage
Text Embedding	LLaMA 3.2 / InstructorXL / BGE /SentenceTransformers / LLaMA tokenizer
Vision Model	CLIP / LLaVA / BLIP-2
Video Processing	FFmpeg, OpenCV
Audio Transcription	Whisper / WhisperX
Audio Embedding	Wav2Vec2 / Transcript-based
Orchestration	LangChain / Haystack / LlamaIndex
LLM	LLaMA 3.2 Vision (multimodal)

6. Evaluation Metrics

Metric	Description
MRR / Precision@k	For ranked search quality
Semantic Relevance Score	Human/automated judgment
Query Response Time	<1 second is typical for top-k retrieval

Modality Coverage	% queries matched from each modality
Feedback Loop	Collect thumbs-up/down from users