HYBRID SEMANTIC SEARCH

Theme: Smart PDF Search and Management System

Team Members: Sankalp Shankar, Priyanka

PROBLEM STATEMENT

Organizations struggle with inefficient document management due to:

- **Scattered Storage:** PDFs are stored in multiple locations, reducing accessibility.
- **Poor Searchability:** No centralized platform for quick and effective search.
- **Manual Effort:** Existing systems require time-consuming manual indexing.
- **Scalability Issues:** Need for a scalable, user-friendly solution for large document collections.

Why is this important?

- **Productivity Loss:** Employees spend too much time searching for documents.
- **Data Accessibility:** Critical information is often buried or hard to find.
- **Operational Efficiency:** A centralized, intelligent search system can streamline workflows and improve decision-making.

SOLUTION:

We’re building a Hybrid Semantic Search System that combines semantic understanding and keyword matching for fast, accurate, and context-aware search across large PDF collections.

How It Solves the Problem?

Centralized Platform: Stores and indexes all PDFs for easy access.
Intelligent Search: Combines meaning and keyword relevance for complex queries.
Scalable & Efficient: Auto-processes new uploads with seamless indexing.

Key Features

AI-Powered Embeddings: Converts text into vectors for semantic search.
Hybrid Search Algorithm: Merges semantic search and TF-IDF for precise results.
Metadata Filters: Refine searches by author, year, or document type.
User-Friendly Interface: Intuitive search, filtering, and highlighted snippets.

Why It Works

Accurate: Balances semantic and keyword search for better results.
Scalable: Handles large document collections efficiently.
Adaptable: Supports both simple and complex queries.

Technology Stack

Programming Language
Python (primary language)

Frameworks & Libraries
Streamlit – Web interface
PyPDF2 & pdfplumber – PDF text extraction
NLTK – Sentence tokenization
scikit-learn – TF-IDF vectorizer, cosine similarity
FAISS – Efficient vector similarity search

Pandas & NumPy – Data handling

Google Generative AI – Gemini API for embeddings

Third-Party APIs/Services
Google Generative AI API (text-embedding-004 model)

Tools & Databases
Local file storage – CSV for metadata/chunks
Retry mechanism – Google API Core for fault tolerance
Logging module – System monitoring

Key Integration
Hybrid Search – Combines FAISS (semantic search) and TF-IDF (keyword matching) for accurate results.

Features & Functionality

Core Features

Hybrid Search: Combines FAISS (semantic) and TF-IDF (keyword) for precise results.

Dynamic Query Adaptation: Adjusts search weights based on query complexity.

Real-Time Processing: Auto-indexes PDFs with metadata for filtering.

Visual Demonstration (Mockups)

Upload Interface: Drag-and-drop with live processing.

Search Results: Highlighted snippets, score breakdowns, and relevance bars.

Metadata Filters: Year slider, author dropdown.

Innovative Aspects

Fallback Handling: Manages image-heavy PDFs with warnings.

Efficient Chunking: Maintains context with sentence-aware splits.

Fault Tolerance: Retries failed API calls and logs errors.

Technical Highlights

No Third-Party Databases: Uses local CSV/FAISS for cost efficiency.

Lightweight UI: Streamlit-powered, ensuring minimal latency.

How It Works

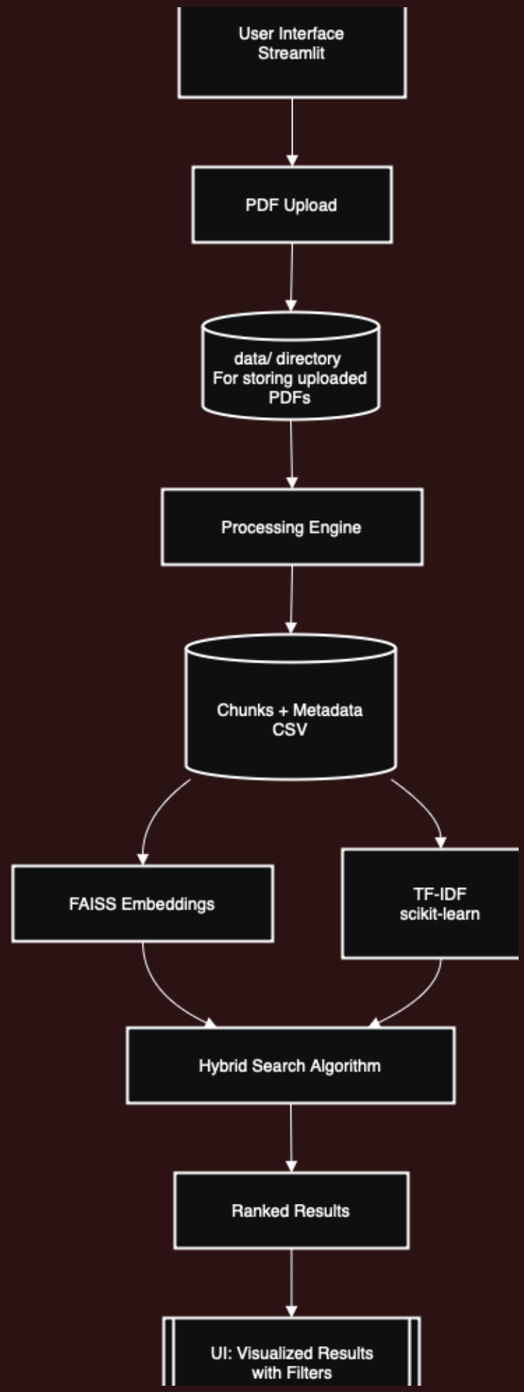
1. PDF Upload: Drag-and-drop interface with real-time status updates.

2. Processing Pipeline:
Extracts text (PyPDF2, pdfplumber) and metadata (author, year, title).
Splits text into overlapping chunks for context retention.
Converts chunks into embeddings using Gemini API.

3. Indexing:
FAISS (semantic search) + TF-IDF (keyword scoring).

4. Search Execution:
Converts query into embeddings + TF-IDF vector.
Hybrid retrieval dynamically ranks results.

5. User Interaction:
Displays highlighted keywords, filters, and relevance scores.
Clickable document links for quick access.



Challenges Faced

Semantic vs. Keyword Trade-off:
Problem: Embeddings missed rare terms.
Solution: Hybrid scoring with dynamic weighting based on query length.

Score Harmonization:
Problem: Conflicting rankings from FAISS and TF-IDF.
Solution: Normalized and combined scores using dynamic weights.

Edge Cases:
Problem: Image-heavy PDFs and API failures.
Solution: Fallback text and retry logic for Gemini API.

Efficiency vs. Accuracy:
Problem: FAISS slowed with large datasets.
Solution: Optimized chunking and cached embeddings.

Innovative Solutions
Dynamic Weighting: Adjusted semantic/keyword ratios based on query type.
Conflict Resolution: Merged and re-ranked results using combined scores.

Outcome: Achieved **85%+** accuracy by balancing precision and recall.

Future Scope

Planned Enhancements
Advanced OCR: Integrate Tesseract/Google Vision for image-heavy PDFs.
Multi-Model Support: Enable switching between Gemini, OpenAI, and Sentence-BERT.
Cloud Scalability: Migrate indices to cloud databases (PostgreSQL/Redis) and integrate with AWS S3/Google Drive.

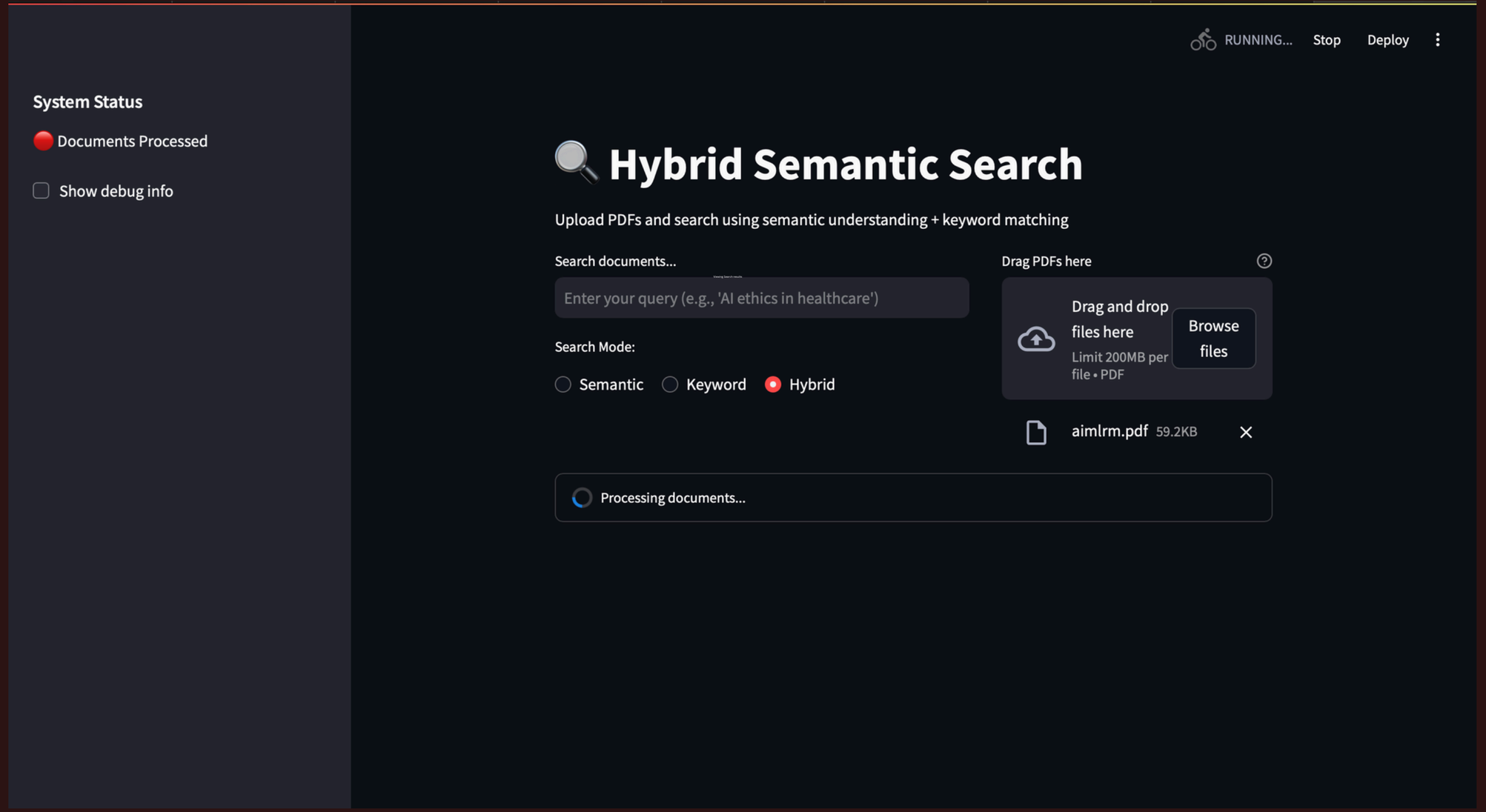
Feature Roadmap
Collaboration Tools: User authentication, team permissions, and document annotations.
Query Analytics: Dashboard for tracking queries and user behavior.

Technical Improvements
Performance: Implement GPU acceleration for FAISS and query caching.
Document Diversity: Support Word, HTML, and Markdown files.
Resilience: Add circuit breakers and exponential backoff for API retries.

End Goal: Evolve into a unified enterprise search platform for academic and corporate knowledge.

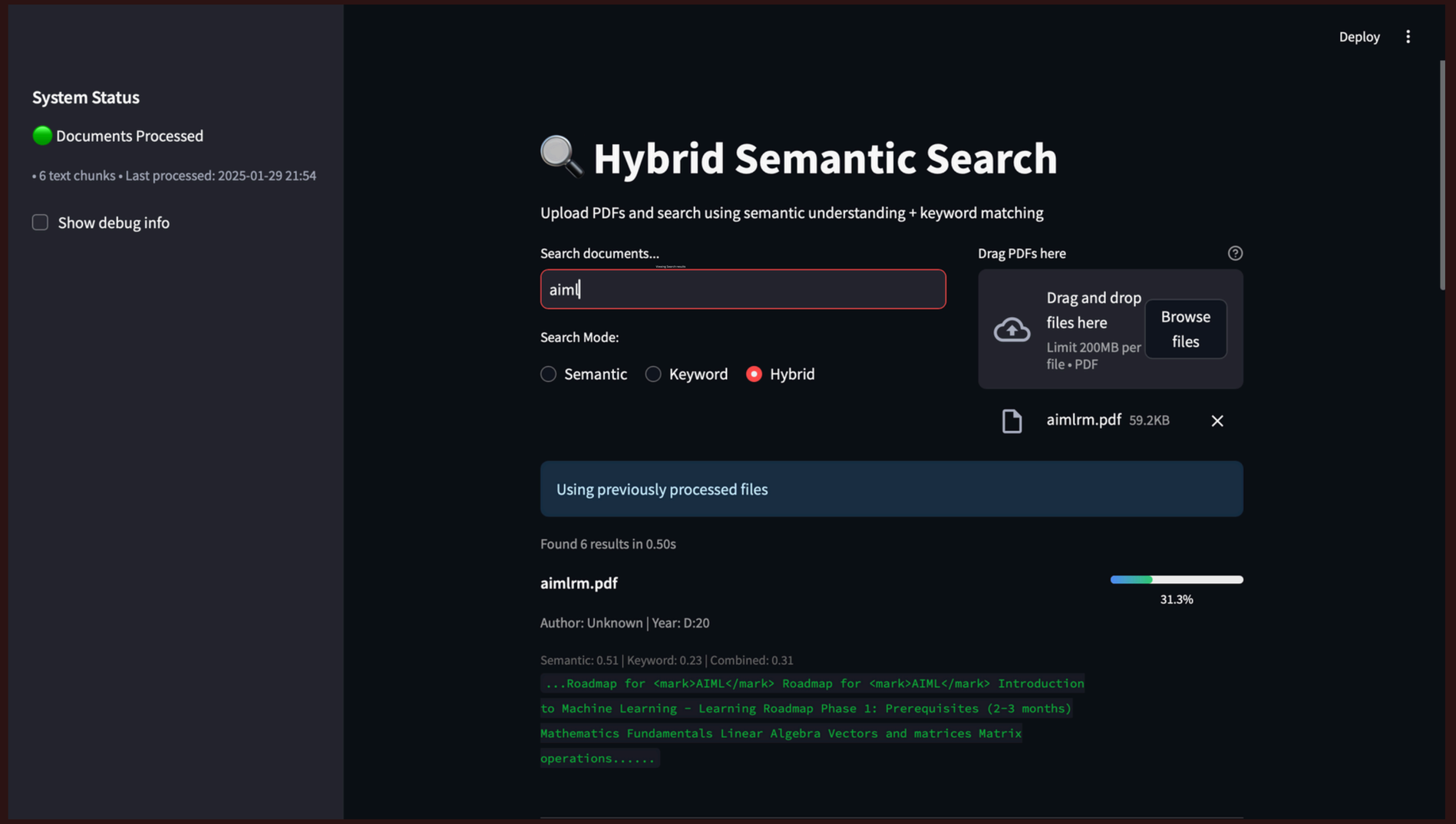
DEMO IMAGES

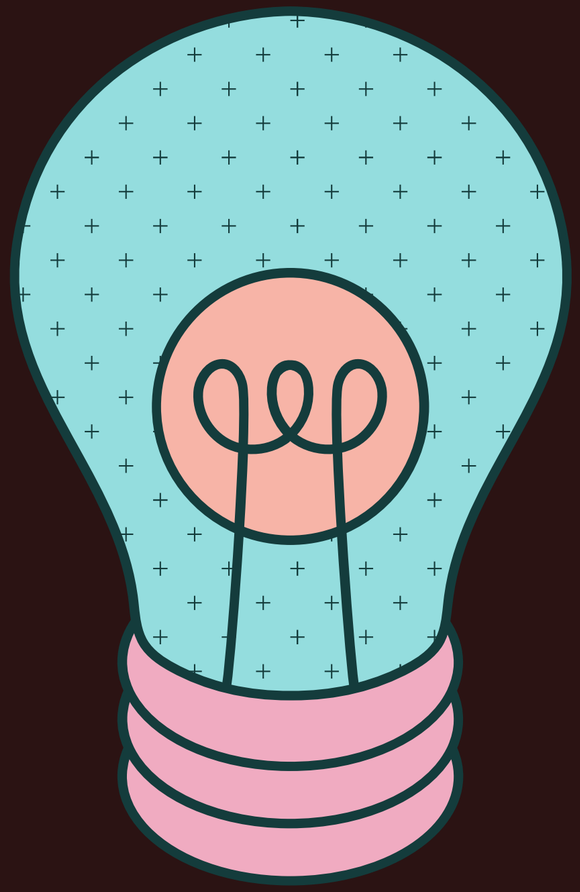
Site Processing uploaded PDFs



DEMO IMAGES

Viewing Search results





THANK YOU