GenAI Sample Project: Research Paper Management & Analysis Intelligence System

Background

You are working as an **AI Engineer at an Academic Technology & Research Intelligence company** that builds platforms for **universities, research labs, think tanks, and R&D departments**. Researchers today are overwhelmed by the rapid growth of scientific literature across domains such as **AI, healthcare, physics, finance, and social sciences**.

Thousands of new papers are published every month on platforms like **arXiv, Semantic Scholar, PubMed, and IEEE Xplore**. Researchers struggle with:

- Discovering relevant papers efficiently
- Keeping track of citations and related work
- Understanding papers quickly without reading them end-to-end
- Identifying emerging research trends across years

Your task is to build a **Research Paper Management & Analysis Intelligence System**—an AI-powered research assistant that helps users **discover, organize, analyze, and interact with research papers** using LLMs, vector search, and intelligent tool integration.

This project closely mirrors **real-world academic search engines and research intelligence tools**, making it highly attractive to **academic tech companies, research organizations, and applied AI teams**.

Objective

Your objective is to design and implement an **end-to-end research intelligence platform** that:

1. Ingests and manages research papers (PDFs and metadata)
2. Automatically generates structured summaries
3. Enables semantic search and question answering across papers
4. Tracks citations and related work
5. Identifies emerging research themes and trends over time
6. Provides a clean researcher-facing UI using **Streamlit**
7. Uses **MCP** to integrate external research tools and metadata sources

By the end of this project, you should demonstrate strong skills in:

- Document AI for long-form technical content
- Retrieval-Augmented Generation (RAG) using FAISS
- LLM orchestration with LangChain
- Research analytics and trend discovery
- Building production-style AI tools for knowledge workers

Data Sources

You may use one or more of the following sources:

1. **Research Paper PDFs**
    - arXiv papers
    - Open-access journal papers
    - Conference papers (NeurIPS, ICML, ACL, CVPR, etc.)
2. **Paper Metadata (CSV / JSON / API responses)**
    - Title
    - Authors
    - Abstract
    - Publication year
    - Venue
    - Keywords / categories
    - Citation count (if available)
3. **Citation Data (Simulated or Extracted)**
    - Reference sections from PDFs
    - Citation relationships between papers
4. **User Notes (Optional)**
    - User-added tags
    - Highlights
    - Reading status (to-read, reading, completed)

Part I: Paper Ingestion & Representation (Python + LangChain)

1. Research Paper Schema & Data Modeling

Design a unified internal representation for a research paper containing:

- `paper_id`
- `title`
- `authors`
- `abstract`
- `full_text`
- `year`
- `venue`
- `keywords`
- `references`
- `citations` (if available)

Task:

Define Python data models (Pydantic or dataclasses) to represent:

- A research paper
- A paper section (e.g., Abstract, Methodology, Results)
- A citation relationship

---

2. PDF Parsing & Section-Level Text Extraction

- Load research paper PDFs programmatically
- Extract text with attention to:
    - Abstract
    - Introduction
    - Methods
    - Experiments / Results
    - Conclusion
    - References
- Remove headers, footers, page numbers, and equations where necessary

Deliverable:

A Python module that converts a research paper PDF into **structured sections** instead of flat text.

---

3. Metadata Enrichment

- Extract metadata automatically where possible:
    - Title and authors from first pages
    - Abstract section
    - Publication year from metadata or filename
- Allow manual metadata editing via UI (optional)

Reflection Question:

Why is section-level structure especially important for academic papers?

Part II: Knowledge Indexing & Semantic Search (FAISS + LangChain)

4. Intelligent Text Chunking Strategy

- Split paper content into chunks using:
    - Section-based chunking (preferred)
    - Or hybrid section + token-length chunking
- Preserve metadata for each chunk:
    - Paper ID
    - Section name
    - Year
    - Keywords

---

5. Building the Vector Store with FAISS

Using **LangChain + FAISS**:

- Choose and configure an embedding model
- Generate embeddings for all paper chunks
- Store embeddings in FAISS with metadata

Deliverable:

A FAISS-backed vector store with functions:

- `index_papers()`
- `semantic_search(query, filters=None, top_k=5)`

---

6. Semantic Paper Discovery

Implement a discovery pipeline that allows users to:

- Search papers by topic using natural language
- Retrieve:
    - Relevant papers
    - Most relevant sections within each paper

Example Queries:

- “Recent work on transformer efficiency”
- “Papers combining reinforcement learning and robotics”
- “Foundational papers on attention mechanisms”

Part III: Summarization & Research Q&A

7. Automatic Paper Summarization

Using LangChain + LLMs, generate:

1. **Short Summary** (5–6 bullets)
2. **Structured Summary**
    - Problem statement
    - Proposed approach
    - Key contributions
    - Results
    - Limitations
3. **Section-wise Summaries** (optional but recommended)

Task:

Design prompts that ensure:

- Technical accuracy
- Neutral, academic tone
- No hallucinated results

---

8. Context-Aware Question Answering (RAG)

Implement a RAG pipeline where users can ask:

- “What problem does this paper solve?”
- “How does this approach differ from prior work?”
- “What datasets are used?”
- “What are the main limitations?”

Pipeline steps:

1. Retrieve relevant paper chunks via FAISS
2. Construct a context-aware prompt
3. Generate grounded answers using LangChain

---

9. Cross-Paper Comparative Q&A

Extend the system to answer comparative questions such as:

- “Compare methods used in Paper A and Paper B”
- “What are different approaches to diffusion models?”

Task:

Retrieve chunks from multiple papers and combine them intelligently in a single prompt.

Part IV: Citation Tracking & MCP Integration

10. Citation Graph Construction

- Parse reference sections to extract cited paper titles
- Build a citation graph:
    - Paper → References
    - Paper → Cited by (if data available or simulated)

Task:

Store citation relationships in a structured format (graph or adjacency lists).

---

11. MCP Tools for External Research Systems

Design **MCP tools** for:

1. **Paper Metadata Lookup Tool**
    - Input: paper title / DOI
    - Output: metadata (year, venue, citation count)
2. **Related Work Discovery Tool**
    - Input: paper ID
    - Output: semantically related papers + citation-based neighbors
3. **Trend Analytics Tool**
    - Input: topic or keyword
    - Output: publication frequency over time + emerging subtopics

Integrate these tools into your LangChain agent so the LLM can:

- Call tools when metadata is missing
- Enrich answers with citation-aware insights

Part V: Research Trend & Insight Analysis

12. Keyword & Topic Aggregation

- Extract key phrases from abstracts and introductions
- Group papers by:
    - Year
    - Topic
    - Venue

Task:

Analyze:

- Topic growth over time
- Shifts in research focus

---

13. Emerging Trend Identification

Implement logic to identify:

- Rapidly increasing keywords
- New methods appearing in recent years
- Cross-domain convergence (e.g., “LLMs + Biology”)

Deliverable:

A function that returns:

- “Emerging topics”
- Supporting evidence (paper counts, example papers)

Part VI: Frontend – Researcher Assistant UI (Streamlit)

14. Streamlit Paper Management Dashboard

Build a Streamlit app with:

- Paper library view:
  - Title, authors, year, venue
- Filters:
  - Year range
  - Topic / keyword
  - Venue

---

15. Interactive Paper Viewer

When a paper is selected, display:

- Abstract
- Auto-generated summary
- Citation information
- Button to ask questions about the paper

---

16. Research Chat Assistant

Add a chat interface where users can:

- Ask questions about a single paper
- Ask questions across the entire library
- Compare multiple papers

Display:

- Answer text
- Source paper names and sections used

---

17. Citation & Trend Visualization Panel

Add Streamlit components to visualize:

- Citation networks (basic graph or table form)
- Topic trends over time (line charts)
- Most influential papers in a topic

Part VII: Evaluation & Reporting

18. Scenario-Based Evaluation

Test your system using scenarios such as:

1. Quickly understanding a new paper
2. Finding related work for a literature review
3. Comparing competing approaches
4. Identifying fast-growing research areas

Record observations for each scenario.

---

19. Quality Assessment

Evaluate:

- Summary accuracy and completeness
- Answer groundedness (no hallucinations)
- Retrieval relevance
- Usability of the Streamlit UI

Provide a short critical analysis:

- Strengths
- Limitations
- Improvements for production deployment

---

20. Final Deliverables

Your final submission should include:

1. **Python backend**
    - PDF parsing and metadata extraction
    - FAISS-based vector index
    - LangChain RAG and summarization pipelines
    - MCP tool integrations
2. **Streamlit frontend**
    - Research paper dashboard
    - Interactive summaries and chat
    - Trend and citation views
3. **Documentation**
    - System architecture diagram
    - Explanation of design decisions
    - Example queries and outputs
    - Evaluation and reflections

Key Skill Outcomes

By completing this project, students will demonstrate:

- Advanced **document intelligence for technical content**
- **Semantic search and RAG** with FAISS
- **LangChain agents** with MCP-based tool usage
- Research-focused **summarization and analytics**
- A production-style **research assistant UI** using Streamlit