Project Report

Project Title: Multi-Agent AI System
Intern Name: Shiv Sunil Kasat

---

1. Abstract

The Multi-Agent AI System is a **Flask-based intelligent platform** designed to handle user queries dynamically by routing them to specialized AI agents. It integrates **PDF-based retrieval**, **real-time web search**, and **academic paper search** via ArXiv, ensuring accurate and contextually relevant responses. The system emphasizes **decision transparency, data privacy, and modularity**, making it suitable for academic, professional, and research environments.

---

2. Introduction

Finding accurate answers in large volumes of data is challenging. Traditional search methods often fail to provide **context-aware insights**.

This system addresses this problem by:

- Using **multiple AI agents** specialized in PDFs, web search, or academic research.
- Automatically determining the **best agent(s) for a query**.
- Ensuring **secure handling of user-uploaded PDFs**.
- Logging all decisions to maintain **auditability and transparency**.

---

3. Objectives

- Develop an **intelligent query routing system** using AI.
- Implement specialized agents: **PDF RAG**, **Web Search**, and **ArXiv Search**.
- Enable **vector-based semantic search** for PDFs using FAISS.
- Build a **user-friendly Flask interface** for uploads and queries.
- Ensure **decision logging, privacy, and safety**.

---

4. System Architecture

Overview: The system consists of four main components coordinated by a **Controller Agent**:

4.1 Controller Agent

- Powered by **Groq LLM (Llama 3.3)**.
- Analyzes user queries and determines which agent(s) should handle the query.
- Logic includes:

1. **Keyword & context analysis:** Detects domain-specific keywords (e.g., “algorithm”, “AI paper”).
2. **Agent prioritization:** If PDFs exist and the query is technical, PDF RAG is prioritized. For general knowledge or current events, Web Search is prioritized. For academic queries, ArXiv is selected.
3. **Forced PDF search:** Overrides routing if the user requests PDF-only results.

- Decision is logged in logs/decisions.json for transparency.

4.2 PDF RAG Agent

- Handles **uploaded PDFs**.
- Uses **FAISS vector DB** with **Sentence-Transformers embeddings**.
- Provides semantic search for domain-specific documents.

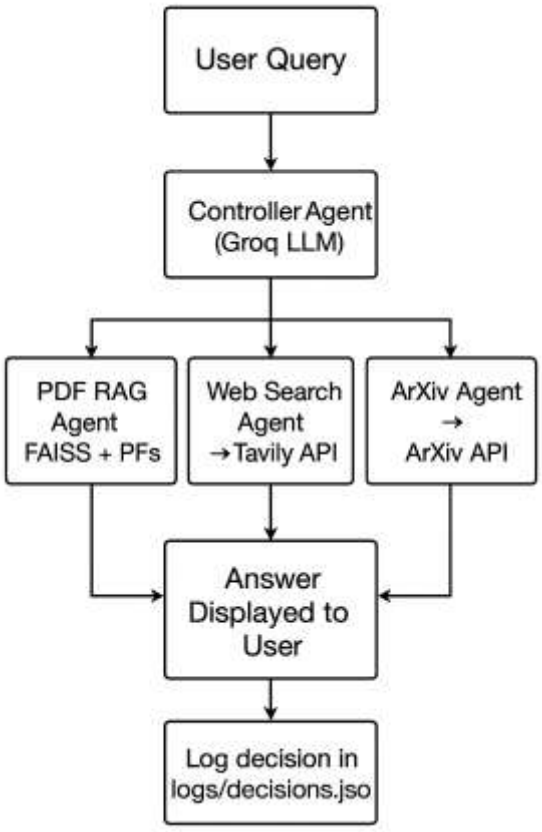
4.3 Web Search Agent

- Performs **real-time searches** using **Tavily API**.
- Retrieves trending and current information.

4.4 ArXiv Agent

- Searches **scientific and academic papers** using **ArXiv API**.
- Ideal for research queries and technical insights.

Flow Diagram:



---

5. Controller Decision Logic

1. **Query Analysis:** The LLM parses the query to identify keywords, question type, and domain.
2. **Agent Scoring:** Each agent is scored for relevance based on query context:
    - PDF RAG: Technical/document-specific
    - Web Search: General knowledge/current events
    - ArXiv: Academic/research papers
3. **Routing:**
    - Highest score agent(s) selected.
    - Forced PDF search bypasses scoring and routes to PDF RAG only.
4. **Logging:** All routing decisions, scores, and user preferences stored for auditing and future improvement.

---

6. Safety and Privacy Handling

- **PDF Security:**
    - Uploaded PDFs stored in a **temporary secure folder (uploads/)**.
    - Access limited to the session to prevent unauthorized access.
- **Data Privacy:**
    - No user data (queries, files) shared with third parties outside the APIs.
    - Decision logs anonymized to prevent user identification.
- **API Safety:**
    - Web and ArXiv API requests sanitized to prevent malicious inputs.
- **Compliance:** Ensures **safe AI usage** by restricting queries to text-based content; no execution of code from uploaded PDFs.

---

7. Features

- **Intelligent Auto-Routing**: Dynamic agent selection based on query type.
- **Force PDF Search:** Restrict query to uploaded documents.
- **Upload & Index PDFs:** Semantic search indexing.
- **Decision Logging:** Tracks all query routing decisions.
- **Preloaded Sample PDFs:** 10 CSE subjects for demonstration.

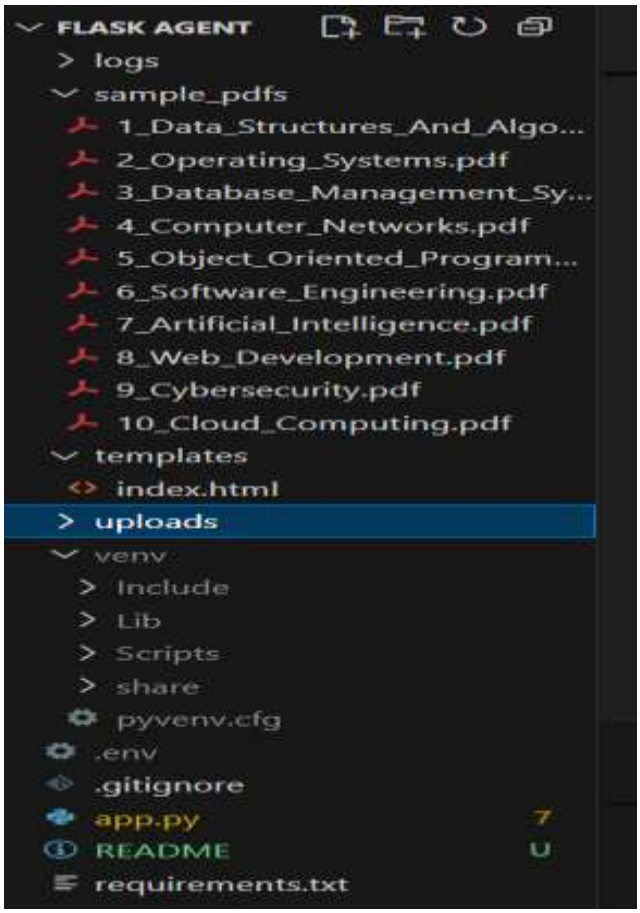
---

8. Tech Stack

Component	Technology / Library
Backend	Flask, Python
LLM	Groq (Llama 3.3)
Vector Database	FAISS
Embeddings	Sentence-Transformers
PDF Processing	PyMuPDF
Web Search	Tavily API
Academic Search	ArXiv API
Frontend	HTML, CSS, Bootstrap

---

9. Project Structure



---

10. Usage Instructions

1. **Upload PDFs:** Choose file → Upload & Index → Wait for confirmation.
2. **Ask Questions:** Examples:
    - "Explain binary search" → PDF RAG
    - "Who is the richest person 2025?" → Web Search
    - "Recent papers on AI" → ArXiv
3. **Force PDF Search:** Check box to search only uploaded PDFs.
4. **View Results:** Answers displayed with decision log entry.

---

11. Limitations

- **Dependency on API availability:** Web search and ArXiv results depend on external APIs.
- **PDF Quality:** Poorly scanned or unstructured PDFs may reduce search accuracy.
- **Query Ambiguity:** Controller may misroute queries with insufficient context.
- **Language Support:** Currently supports only English text in PDFs and queries.
- **Scalability:** Large-scale PDF uploads may require optimized indexing or distributed vector DB.

---

12. Conclusion

The Multi-Agent AI System demonstrates an **intelligent, modular, and secure approach** to handling diverse queries. It combines PDF retrieval, real-time web search, and academic paper search into a **single, user-friendly platform**. The system can serve as a foundation for **research assistants, educational tools, and AI-powered knowledge management systems**.