■ Reckonsys Gen AI Dev Assignment: Agentic RAG Challenge

Objective: Build a basic agentic RAG system where the agent autonomously decides whether to retrieve answers from documents or search the web when confidence is low.

■■ Functional Requirements

1. Document Ingestion: Load and index 5–10 text or PDF documents using embeddings + vector store (FAISS, Chroma, or Milvus). Include metadata like title and source.

2. Retriever-based QA: Implement RAG: query → retrieve → generate. Include confidence scoring based on similarity or LLM evaluation.

3. Agentic Decision Layer: Agent decides whether to return RAG output or trigger web search if confidence < threshold.

4. Web Search Agent: Fetch and summarize information from web using DuckDuckGo, Tavily, or SerpAPI. Include note: "■ This answer was found on the web."

5. Output Format: Return JSON: {"source": "document" | "web", "answer": "string", "confidence": "float", "context_snippets": ["string"]}

■ Tech Expectations

• Backend: Python (FastAPI or Streamlit demo preferred)
• LLM: OpenAI GPT-4o-mini / GPT-4-Turbo or open-source
• Libraries: LangChain, LlamaIndex, FastMCP, Haystack
• Vector DB: FAISS / Chroma / Pinecone
• Search: Tavily / DuckDuckGo / SerpAPI

■ Bonus Points

• Implement ReAct-style reasoning or tool-calling flow.
• Add UI or CLI interface.
• Include logs or reasoning trace.
• Deploy via FastAPI or Gradio.
• Provide README.md with architecture overview.

■ Submission Deliverables

1. GitHub repo with code, docs, and sample docs folder.
2. Optional demo video (≤ 2 mins).
3. Deadline: 72 hours from assignment start.

■ Evaluation Criteria

Criteria	Weight
Architecture clarity	25%
RAG setup & performance	25%
Agentic logic implementation	25%
Code quality & documentation	15%
Creativity / Bonus features	10%