# PDF Question Answering System (RAG)

## 1. Problem Understanding & Objective

- Identified the challenge of extracting meaningful information from large PDF documents using traditional keyword search.
- Designed a **Retrieval-Augmented Generation (RAG)** system to enable natural-language querying over document content.
- Focused on building a solution that is **privacy-preserving**, **scalable**, and **safe for public multi-user access**.

## 2. System Design & Architecture

- Implemented a **modular and production-ready architecture** separating concerns across:
  - User Interface (Streamlit)
  - Document ingestion and preprocessing
  - Embedding and vector storage
  - Retrieval and LLM-based reasoning
- Designed the system to operate **entirely in memory**, avoiding any form of disk persistence.
- Ensured session-level isolation so that each user interacts only with their own uploaded document.

## 3. Technical Implementation

- **PDF Processing:**
  - PDFs are read as raw bytes and converted into page-wise textual content.
- **Chunking Strategy:**
  - Used recursive character-based chunking to balance context coverage and retrieval accuracy.
- **Embedding Generation:**
  - Leveraged **BAAI/bge-small-en-v1.5** embeddings for robust semantic representation.
- **Vector Storage:**
  - Utilized **ChromaDB** in RAM-only mode for efficient similarity search.
- **Retrieval & Answer Generation:**
  - Integrated **Groq's Llama 3.3 70B** model to generate accurate, context-grounded responses.
  - Displayed source references to enhance transparency and explainability.

## 4. State Management & Reliability

- Implemented automatic session resets when a new PDF is uploaded.
- Ensured vectorstores and embeddings are re-initialized per session to maintain correctness.
- Achieved reliable behavior across multiple uploads and concurrent users through careful session handling.

## 5. Tools & Technologies Used

- **Language:** Python
- **Frontend:** Streamlit
- **LLM:** Groq Llama 3.3 70B
- **Embeddings:** BAAI/bge-small-en-v1.5
- **Vector Database:** ChromaDB (in-memory)
- **Frameworks:** LangChain, Hugging Face

## 6. Key Learnings & Skills Gained

- Gained hands-on experience with **end-to-end RAG pipeline development**.

- Learned the importance of **vectorstore lifecycle management** in dynamic applications.
- Developed strong understanding of **session-based state handling** in interactive AI systems.
- Improved ability to design **secure, privacy-first AI architectures**.
- Strengthened skills in **debugging real-world ML integration issues** involving embeddings and retrieval.
- Enhanced knowledge of deploying **high-performance LLMs** for real-time applications.