🆓 Build Your First LLM Application - Raw and Stupid!

A completely free document Q&A application with conversation memory, built during the AI Summer of Code Season 2.

# 🎯 What This App Does

Upload any PDF document and have an intelligent conversation about it! The AI remembers your previous questions and can reference them naturally.

Example conversation:

**You:** "What is this research paper about?"  
**AI:** "This paper discusses machine learning applications in healthcare..."  
  
**You:** "What methodology did they use for that?"  
**AI:** "For the machine learning applications in healthcare that I mentioned, they used..."  
  
**You:** "How does it compare to previous work?"  
**AI:** "Compared to the previous work mentioned in the paper..."

# 💰 Cost Breakdown

**Total cost: $0.00**

* ✅ Groq API: Free tier (thousands of requests/day)
* ✅ HuggingFace Embeddings: Run locally (completely free)
* ✅ Streamlit: Free framework
* ✅ FAISS Vector Search: Free, runs locally
* ✅ Streamlit Cloud Hosting: Free

No credit card required. No hidden fees. Actually free.

# 🛠️ Technology Stack

**Groq:** Lightning-fast LLM inference (< 1 second response times) (https://groq.com/)

**HuggingFace:** Local embeddings with all-MiniLM-L6-v2 (https://huggingface.co/)

**Streamlit:** Web framework for the UI (https://streamlit.io/)

**LangChain:** Document processing and text splitting (https://langchain.com/)

**FAISS:** Fast similarity search for vectors (https://faiss.ai/)

# 🏃‍♂️ Quick Start

## 1. Clone the Repository

git clone https://github.com/aisummerofcode/aisoc-season-2.git  
cd "Applied AI/src/week\_1/day\_3\_first\_llm"

## 2. Install Dependencies

pip install -r requirements.txt

## 3. Get Your Free Groq API Key

1. 1. Go to console.groq.com
2. 2. Sign up (no credit card required)
3. 3. Create an API key

## 4. Run the Application

streamlit run app.py

## 5. Start Chatting!

1. 1. Enter your Groq API key in the sidebar
2. 2. Upload a PDF document
3. 3. Ask questions and have a conversation!

# 🧠 How It Works (The RAG Pattern)

This app implements RAG (Retrieval-Augmented Generation) with conversation memory:

1. 1. Document Processing: PDF is split into chunks with overlap
2. 2. Local Embeddings: Each chunk gets an embedding vector (runs on your computer)
3. 3. Vector Storage: FAISS index for fast similarity search
4. 4. Question Processing: Find relevant chunks for each question
5. 5. Conversation Context: Include previous Q&A pairs as context
6. 6. LLM Generation: Groq generates answers using document context + conversation history

# 🎛️ Available Models

**llama-3.1-8b-instant (Default):** Fast and smart, 131k context window

**llama-3.3-70b-versatile:** Most capable, 131k context window

**gemma2-9b-it:** Balanced option, 8k context window

# 🔧 Configuration Options

## Chunk Settings

* Chunk Size: 1000 characters (adjustable)
* Overlap: 200 characters (prevents information splitting)
* Retrieval: Top 4 most relevant chunks per question

## Conversation Memory

* History Length: Configurable (3-20 exchanges)
* Token Management: Automatic trimming to prevent overflow
* Context Awareness: References previous answers naturally

## Model Parameters

* Temperature: 0.1 (factual, consistent answers)
* Max Tokens: 1000 (reasonable response length)
* System Prompt: Optimized for document grounding + conversation

# 📁 Project Structure

day\_3\_first\_llm/  
├── app.py # Main application  
├── requirements.txt # Python dependencies  
├── README.md # This file  
├── slides/ # Presentation slides  
│ └── slides.md  
└── examples/ # Example PDFs for testing  
 └── sample\_document.pdf

# 🚀 Deployment

## Deploy to Streamlit Cloud (Free)

### 1. Push to GitHub:

git add .  
git commit -m "Add LLM app"  
git push origin main

### 2. Deploy:

* Go to share.streamlit.io
* Connect your GitHub repository
* Select this folder as the app directory
* Deploy with one click!

# 🔍 Troubleshooting

## Common Issues

### "No relevant information found"

* Document might be scanned (needs OCR)
* Try rephrasing your question
* Check if PDF text extracted properly

### Rate limiting errors

* Free tier has generous limits but not unlimited
* Wait a moment and try again
* Consider upgrading for heavy usage

### Wrong or incomplete answers

* Check the "View source chunks" section
* Information might be split across chunks
* Try adjusting chunk size or asking more specific questions

### Conversation context issues

* Clear conversation history if it gets too long
* Reduce max history length in sidebar
* Be specific when referencing previous answers

# 🎓 Learning Objectives

By building this app, you learn:

* RAG Pattern: The foundation of most AI applications
* Vector Databases: How similarity search works
* API Integration: Working with modern AI APIs
* Conversation Management: Maintaining context across exchanges
* Cost Optimization: Building powerful apps for free
* Production Deployment: Getting your app online

# 🔮 Extension Ideas

## Easy Additions

* ☐ Export conversations to PDF
* ☐ Multiple document support
* ☐ Custom CSS for better UI
* ☐ Question suggestions based on document content

## Advanced Features

* ☐ User authentication
* ☐ Document management system
* ☐ API endpoints for integration
* ☐ Custom embedding models for specific domains
* ☐ Conversation summarization
* ☐ Multi-language support

# 📚 Resources

**Groq Documentation:** console.groq.com/docs

**Streamlit Docs:** docs.streamlit.io

**LangChain Guide:** python.langchain.com

**HuggingFace Models:** huggingface.co/models

# 🤝 Contributing

This is part of the AI Summer of Code Season 2 curriculum. Feel free to:

* Report bugs or issues
* Suggest improvements
* Add new features
* Share your extensions

# 📄 License

MIT License - feel free to use this code for learning and building!

# 🙏 Acknowledgments

* AI Summer of Code for the amazing learning opportunity
* Groq for providing free, fast LLM inference
* HuggingFace for open-source embeddings
* Streamlit for making web apps simple

Built with ❤️ during AI Summer of Code Season 2

Remember: The best AI application is the one that actually gets used. Start simple, ship fast, iterate based on feedback!