# Chat with Multiple PDFs

**Project Report submitted in the partial fulfilment**
**of**
Bachelor of Technology
In
Computer Engineering


**SVKM's NMIMS University**
(Deemed-to-be University)


**School of Technology Management and Engineering(STME)**
**Indore, Madhya Pradesh**

**2025-2026**


**Submitted By: Garv Gangwani**

**Submitted To: Dr. Raj Gaurav Mishra**


**Course:** Natural Language Processing (NLP)

**Project Type:** Mini Project – Individual

**Submission Date:** 30th October 2025

# 1. Introduction

## Project Overview

An intelligent document analysis system that enables users to interact with multiple PDF documents through natural language conversations. Users can upload PDFs, ask questions, and receive accurate, context-aware responses powered by AI.

## Objectives

- Enable conversational interaction with PDF documents
- Support multiple document uploads and unified search
- Provide fast, accurate responses using semantic search
- Maintain conversation context for follow-up questions

---

# 2. Technology Stack

| Component | Technology |
|---|---|
| Frontend | Streamlit |
| PDF Processing | PyPDF2 |
| Text Processing | LangChain |
| Embeddings | Hugging Face (sentence-transformers) |
| Vector Database | FAISS |
| Language Model | Groq API (Llama/Mixtral) |
| Memory | ConversationBufferMemory |

---

# 3. System Architecture

The application uses a **Retrieval Augmented Generation (RAG)** architecture:

1. **Document Processing**: Extract and chunk text from PDFs
2. **Embedding Generation**: Convert text to vectors using local embeddings
3. **Vector Storage**: Store in FAISS database for fast similarity search

4. **Query Processing**: Search relevant chunks based on user questions
5. **Response Generation**: Use Groq LLM to generate contextual answers
6. **Display**: Show conversation in styled chat interface

**Data Flow**:

PDF Upload → Text Extraction → Chunking → Embeddings → Vector DB →
User Query → Similarity Search → Context Retrieval → LLM → Response

---

# 4. Key Features

## 4.1 Multi-Document Support

- Upload and process multiple PDFs simultaneously
- Unified knowledge base across all documents
- Cross-document information retrieval

## 4.2 Semantic Search

- Understands query intent beyond keywords
- Retrieves contextually relevant information
- Handles synonyms and related concepts

## 4.3 Conversational AI

- Maintains conversation history
- Understands follow-up questions
- Context-aware responses

## 4.4 Robust Design

- Multiple model fallback mechanism
- Efficient local embedding generation (no API costs)
- Error handling with clear user feedback

---

# 5. Implementation Details

## Document Processing

- **Text Extraction**: PyPDF2 extracts text from all pages
- **Chunking**: 1000 characters per chunk with 200 character overlap
- **Purpose**: Maintains context while enabling efficient search

### Embeddings & Vector Storage

- **Model**: sentence-transformers/all-MiniLM-L6-v2 (384 dimensions)
- **Storage**: FAISS vector database for fast similarity search
- **Advantage**: Runs locally, no API calls required

### Conversational Chain

- **Models**: llama-3.1-8b-instant (primary), with 3 fallback options
- **Configuration**: Temperature=0, Max tokens=2048, Retrieves top 3 chunks
- **Memory**: Full conversation history for context

---

## 6. Technical Challenges Solved

| Challenge | Solution |
|---|---|
| Model deprecation | Multi-model fallback with availability testing |
| Large document processing | Text chunking with overlap, cached vector storage |
| Context preservation | ConversationBufferMemory for chat history |
| Response accuracy | Semantic search + top-3 chunk retrieval |

---

## 7. Installation & Usage

### Setup

```
# Install dependencies
pip install streamlit python-dotenv PyPDF2 langchain langchain-groq faiss-cpu
sentence-transformers

# Configure API key in .env
GROQ_API_KEY=your_key_here

# Run application
```

streamlit run app.py

### Usage

1. Upload PDF files via sidebar
2. Click "Process" to analyze documents
3. Ask questions in natural language
4. Receive AI-generated answers with conversation context

---

# 8. Performance

- **Processing**: 1-2 seconds per document
- **Query Response**: 2-4 seconds average
- **Scalability**: Tested with 10+ documents
- **Accuracy**: High precision for relevant information retrieval

---

# 9. Use Cases

**Academic Research**: Literature review, paper analysis, cross-referencing studies

**Professional**: Contract analysis, technical documentation, report insights

**Educational**: Study material comprehension, textbook Q&A

**Business**: Market research analysis, policy queries

---

# 10. Limitations & Future Enhancements

### Current Limitations

- Text-only PDFs (no OCR for scanned documents)
- No image/chart analysis
- Session-based (no persistent storage)
- English language optimized

### Planned Improvements

- OCR support for scanned PDFs
- Citation with source page numbers
- Conversation export functionality
- Support for DOCX, TXT formats
- Document summarization feature
- Persistent storage and multi-user support

---

## 11. Security & Privacy

- PDFs processed locally during extraction
- Embeddings generated without external API calls
- Only queries sent to Groq API
- No permanent document storage
- API keys secured in environment variables

---

## 12. Conclusion

This application successfully demonstrates practical implementation of RAG technology for document analysis. It combines semantic search, vector databases, and large language models to provide an efficient, conversational interface for PDF interaction. The system significantly reduces information extraction time, making it valuable for researchers, professionals, and students.

**Key Achievements**: Functional document Q&A system, efficient semantic search, intuitive interface, robust error handling, optimized performance

---

## System Requirements

**Software**: Python 3.8+, Modern web browser

**Hardware**: 4GB RAM minimum (8GB recommended), 2GB disk space

**API**: Groq API key (free tier available)