C.V. RAMAN GLOBAL UNIVERSITY BHUBANESWAR, ODISHA

LLM-Based Custom Chatbot on AI&DS

Under the guidance of
MR. Arib Nawal
Trainer

SUBMITTED BY:

NAME	REGD NO.
Sai Shiva	CL2025010601902684
Jagjeet Moharana	CL2025010601900462
Shreyasi Utthasanee	CL2025010601962876
Bhumika Rani Das	CL202501060186964
Sumitra Pradhan	CL2025010601870614
Soubhagya Kumar Pradhan	CL2025010601883012

DEPARTMENT OF COMPUTER SCIENCE &

ENGINEERING

C.V. RAMAN GLOBAL UNIVERSITY BHUBANESWAR, ODISHA

CONTENTS

1. **Introduction**
   - 1.1 Abstract
   - 1.2 Problem statement
   - 1.3 objective
2. **Literature Review**
3. **System Requirements**
4. **System Design**
   - 3.1 Architecture
   - 3.2 Flowchart
5. **Implementation**
   - 4.2 Code overview
   - 4.1 Model demo screenshot
6. **Results and Evaluation**
7. **Applications**
8. **Limitations**
9. **Future Enhancements**
10. **Conclusion**
11. **References**

1. INTRODUCTION

1.1 Abstract

This project focuses on creating a custom chatbot that can answer questions from user-uploaded documents like PDFs, and text files. The chatbot works offline and uses a local Large Language Model (LLM) along with Retrieval-Augmented Generation (RAG) to understand and respond accurately. It uses tools like FAISS for searching through data, and Streamlit to create a simple and interactive user interface. The main goal is to help users get answers directly from their own documents without needing internet or exposing private data to third-party services.

1.2 Problem Statement

People often struggle to find specific information from large documents. Existing tools may need the internet or send data to external servers, which can cause privacy issues. Also, many tools don't support multiple file types or cannot read scanned documents properly. Therefore, there is a need for a simple, secure, and smart chatbot that:

- Works offline.
- Reads and understands various document types.
- Gives fast and accurate answers based on the user's questions.

1.3 Objective

The primary objective of this project is to develop a local, intelligent document chatbot that allows users to ask questions and receive relevant information extracted from their documents. The key objectives include:

1. Support Multiple Formats: Allow users to upload and process documents in PDF, and text files.
2. Efficient Document Indexing: Use FAISS for fast and accurate similarity-based search over extracted content.
3. Local Query Processing: Employ a local LLM (LLaMA model) to answer user queries without needing an internet connection.
4. Real-Time Responses: Stream answers to the users in real-time, making the interaction smooth and responsive.
5. Data Privacy: Ensure complete local data processing to maintain user privacy and security.

2. LITERATURE REVIEW

In recent years, the rapid advancement of Large Language Models (LLMs) such as GPT, BERT, and LLaMA has significantly transformed the landscape of Natural Language Processing (NLP). These models have set new standards in tasks such as question answering, summarization, and conversational AI by enabling machines to generate coherent, human-like responses.

To enhance the accuracy and relevance of chatbot systems, the Retrieval-Augmented Generation (RAG) framework has emerged as a powerful paradigm. RAG enhances traditional generative models by incorporating an external knowledge retrieval mechanism. Instead of relying solely on pre-trained data, RAG-based systems first retrieve relevant information from a user-provided document set and then generate context-aware responses based on the retrieved content. This results in answers that are both precise and grounded in source documents.

Several key technologies and frameworks support the development of such systems:

- FAISS (Facebook AI Similarity Search): An efficient indexing and retrieval library that enables fast semantic search across document chunks.
- Sentence-Transformers: Used to convert text into high-dimensional embeddings that preserve semantic similarity, improving retrieval quality.
- LangChain: A robust framework for integrating LLMs with document loaders, vector stores, conversational memory, and custom retrieval pipelines.
- llama.cpp: A lightweight, CPU-compatible implementation of LLaMA models that supports local inference, enabling chatbot deployment without internet access or GPU resources.

Earlier systems such as Dense Passage Retrieval (DPR) and T5 demonstrated the feasibility of dense embeddings and transformer-based architectures for question answering. However, these approaches often relied on cloud-based APIs, making them unsuitable for scenarios requiring offline access or strict data privacy.

Our current project builds on these innovations by implementing a fully offline, document-aware custom chatbot. It allows users to upload documents in various formats (PDFs, text files), performs semantic indexing using FAISS and Sentence-Transformers, and uses LLaMA models via llama.cpp for efficient, private, and real-time response generation. The integration with LangChain ensures modularity and ease of interaction across components.

3. SYSTEM REQUIREMENTS

Software Requirements:

- OS: Windows/Linux/macOS
- Python 3.8 or above
- Required Python packages:
    - streamlit
    - faiss-cpu
    - sentence-transformers
    - llama-cpp-python
    - langchain
    - langchain-community
    - PyPDF (for PDF parsing)
    - Asyncio

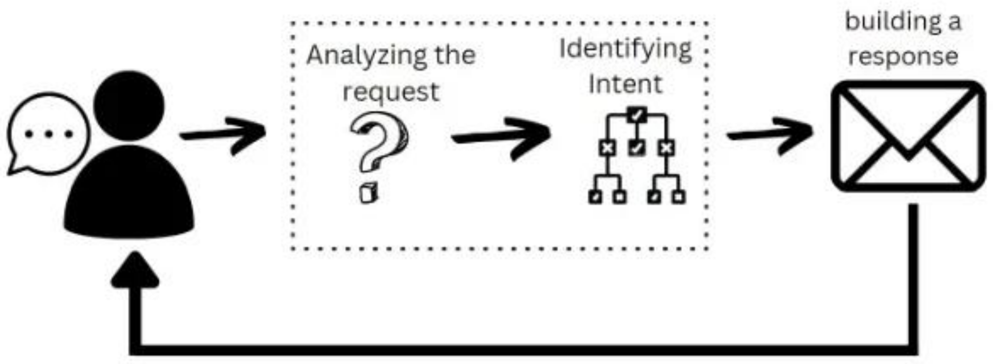


Figure 1: How a chatbot works

4. SYSTEM DESIGN

4.1 Architecture Overview:

1. User Interaction:
   - The user interacts with the system through a Streamlit web interface. The user can upload documents (PDF, TXT).
2. File Handling and Processing:
   - The LocalRAGChatbot class handles document processing. It checks the file types and processes them accordingly:
     - Text files (TXT) and PDF files are directly loaded and parsed.
3. Text Chunking and Embedding:
   - The loaded documents are split into smaller text chunks using RecursiveCharacterTextSplitter. This helps in handling large documents by breaking them down into smaller, more manageable pieces.
   - These chunks are then embedded into vectors using the SentenceTransformer model (all-MiniLM-L6-v2). These embeddings are stored in a FAISS index for efficient retrieval during querying.
4. Search & Retrieval:
   - When the user asks a question, the query is embedded into a vector using the same SentenceTransformer model. This query vector is then used to search the FAISS index for the most relevant document chunks.
5. LLM-based Answer Generation:
   - The top results (document chunks) retrieved from the FAISS index are then used to build a prompt for the LLaMA-based model. The LLaMA model generates an answer based on the provided context.
6. Streaming the Response:
   - The generated answer is streamed back to the user token-by-token for a more interactive experience.

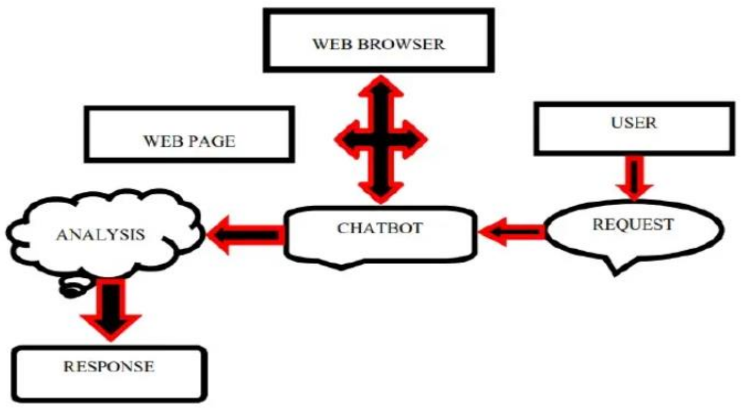
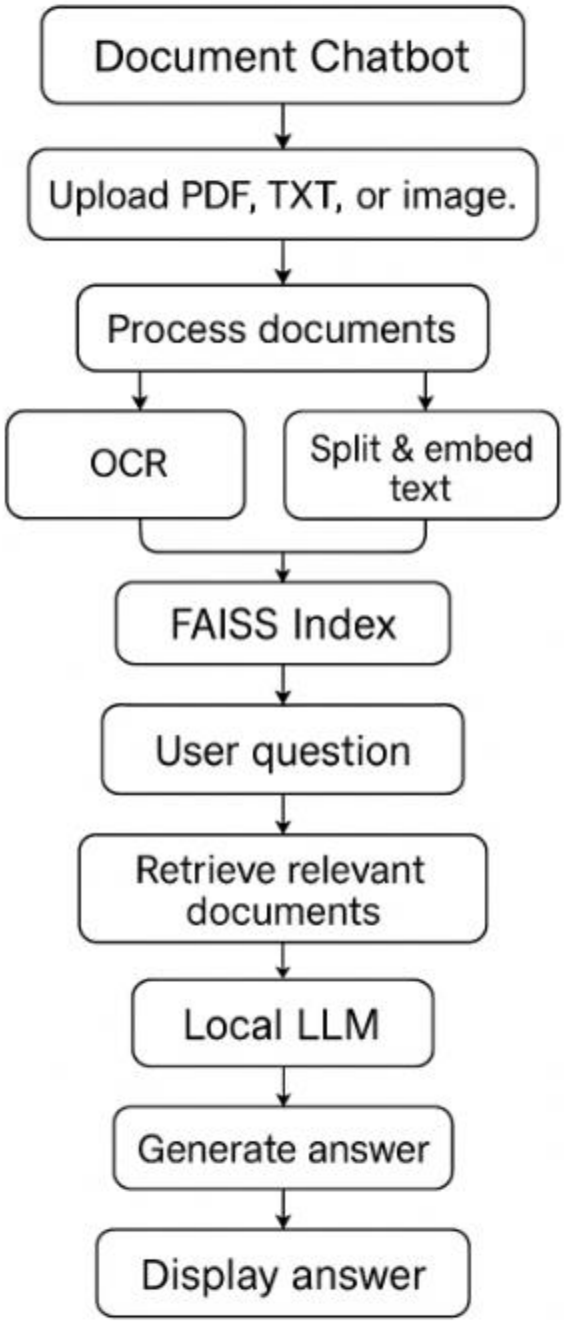


Figure2. System Architecture

4.2 Workflow Diagram



5. IMPLEMENTATION

5.1 Code Overview

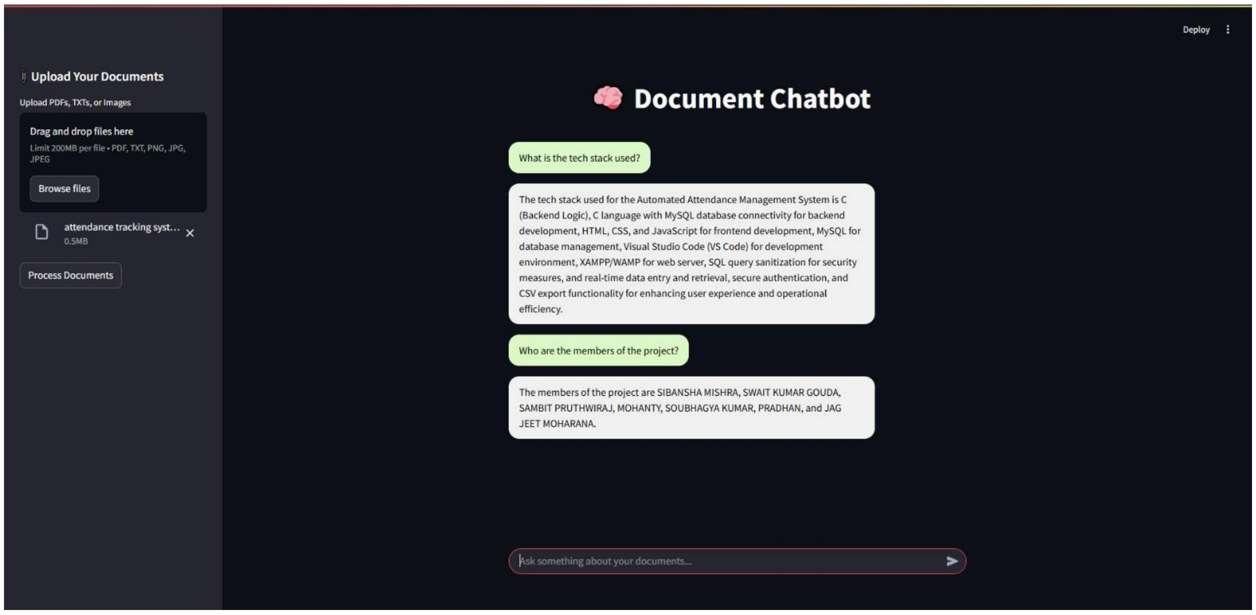
Frontend (app.py):

- Built using Streamlit to provide an interactive user interface.
- Supports file upload functionality for PDFs, text files.
- User input for queries is captured, and responses are displayed in a chat format with dynamic bubbles using Markdown.

Backend (chatbot.py):

- The LocalRAGChatbot class handles the core functionality of the chatbot.
- SentenceTransformer is used to convert text documents into vectors for semantic understanding.
- FAISS is employed for efficient similarity search to find relevant document chunks based on user queries.
- The LLaMA model (via llama-cpp-python) is leveraged for response generation to generate meaningful and contextually relevant answers.

5.2 Demo model screenshot


Deploy
Upload Your Documents
Upload PDFs, TXTs, or Images
Drag and drop files here
Limit 200MB per file • PDF, TXT, PNG, JPG, JPEG
Browse files
attendance tracking syst...
0.5MB
Process Documents
Document Chatbot
Ask something about your documents...

What is the tech stack used?

The tech stack used for the Automated Attendance Management System is C (Backend Logic), C language with MySQL database connectivity for backend development, HTML, CSS, and JavaScript for frontend development, MySQL for database management, Visual Studio Code (VS Code) for development environment, XAMPP/WAMP for web server, SQL query sanitization for security measures, and real-time data entry and retrieval, secure authentication, and CSV export functionality for enhancing user experience and operational efficiency.

Who are the members of the project?

The members of the project are SIBANSHA MISHRA, SWAIT KUMAR GOUDA, SAMBIT PRUTHWIRAJ, MOHANTY, SOUBHAGYA KUMAR, PRADHAN, and JAG JEET MOHARANA.

6. RESULTS AND EVALUATION

The performance of the **LLM-Based Custom Chatbot** was evaluated based on accuracy, response time, usability, and its ability to operate fully in an offline environment. The chatbot was tested with various types of documents and user queries to ensure reliable performance under realistic use-case scenarios.

a. Functional Testing

Sample Test Case:

- Input Document: Resume PDF
- User Query: "List all the skills mentioned in the resume."
- Output: "The key skills include Python, SQL, data analysis, and machine learning."
- Evaluation: The response was accurate and contextually relevant, demonstrating the chatbot's ability to extract information from a variety of document types.

b. Response Time

- Average response time: – 10 to 20 seconds on a GPU-powered system.
- Despite utilizing powerful GPU acceleration, the response time for processing queries is between 10 to 20 seconds. This time includes document processing, chunk retrieval, and inference generation. The GPU helps to speed up embedding, indexing, and inference but the model's size and complexity, along with document processing steps like OCR, still contribute to a few seconds of delay.

c. Accuracy and Relevance

- The system demonstrated high semantic accuracy due to effective SentenceTransformer embeddings and relevant document chunk retrieval using FAISS.
- Contextual relevance in the answers was ensured by the thoughtful construction of prompts and the use of the LLaMA model for local response generation. The system was able to generate responses that were both accurate and based on the specific content within the uploaded documents.

d. Offline Capability

- The chatbot operates fully offline by utilizing local models for both document embedding and response generation.
- During testing, the system successfully functioned with the internet disabled, confirming that both document indexing and model inference were done entirely on local hardware without any reliance on external APIs or cloud-based services.

e. Usability

- The Streamlit-based user interface is intuitive, responsive, and easy to use.
- The system supports drag-and-drop uploads of .pdf, .txt files, making document input straightforward.
- Outputs are displayed in a clean, markdown-friendly format, ensuring an easy-to-read user experience. The interface also handles continuous streaming of responses, providing a smooth interaction with the chatbot.

7. APPLICATIONS

The LLM-Based Custom Chatbot, designed to run entirely offline while processing user-uploaded documents, has a wide range of practical applications across various industries and domains. Its capability to extract, understand, and respond to content-specific queries makes it highly versatile:

1. Academic and Research Assistance

- Helps students and researchers query long research papers, journals, and theses to extract key information instantly.
- Facilitates document summarization, keyword extraction, and Q&A from lecture notes and textbooks.

2. Legal Document Analysis

- Enables offline review and interrogation of legal contracts, agreements, and policy documents.
- Assists legal professionals in quickly finding clauses, terms, and obligations without manual search.

3. Healthcare and Medical Records

- Assists doctors and administrators in extracting relevant patient data from reports and prescriptions.
- Can support hospital record analysis while maintaining strict privacy through offline operation.

4. Corporate Knowledge Base Assistant

- Employees can query internal company handbooks, training manuals, and policy documents without relying on cloud systems.
- Increases productivity by reducing the need to manually search through documents.

5. Government and Administrative Use

- Supports local government offices by enabling data extraction from public notices, circulars, and internal memos in an offline, secure environment.
- Ideal for remote regions with limited internet connectivity.

6. Confidential Data Environments

- Perfect for military, research labs, and private enterprises where sensitive data cannot be uploaded to the cloud.
- Maintains full data confidentiality while enabling intelligent document interaction.

8. LIMITATION

Despite the improved capabilities of the LLM-Based Custom Chatbot, some limitations remain that can be addressed in future development:

1. **Limited Multimodal Support**
   • The chatbot supports text-based documents (PDF, TXT) and basic image-to-text extraction from images (JPG) using OCR. However, it does not yet support audio, video, or scanned handwritten documents effectively.
2. **Hardware Dependency**
   • Although the current model leverages GPU acceleration for faster processing, its performance may vary depending on the user's hardware configuration. On low-end systems, even with GPU, large document processing may still face slight latency.
3. **English Language Only**
   • The system currently processes only English language documents and queries. Multilingual support is not yet implemented.
4. **Limited Context Window**
   • Due to model constraints and token limits, only a limited amount of relevant context can be retrieved and processed at once. This may lead to less comprehensive answers for lengthy or complex documents.
5. **Single-User Interaction**
   • The application is designed for local, individual use and does not currently support concurrent multi-user access or session handling.
6. **Basic UI Features**
   • While the Streamlit-based UI includes support for drag-and-drop uploads, markdown output, and a responsive layout, it lacks advanced features such as:
      - Persistent chat history
      - Voice input or speech recognition
      - Answer highlighting within documents

9. FUTURE ENHANCEMENT

While the current implementation of the LLM-based Custom Chatbot demonstrates robust performance in offline document-based question answering, several enhancements can further elevate its capability, performance, and user experience. The following future improvements are proposed:

1. **Multilingual and Regional Language Support**
   To make the chatbot accessible to a broader audience, support for multiple languages—including regional and non-English languages—can be added. This will enable more inclusive interaction for diverse user demographics.
2. **Integration of Voice-Based Interaction**
   Incorporating speech-to-text for input and text-to-speech for output will allow users to interact through natural voice-based conversation. This is particularly beneficial for visually impaired users or hands-free operation scenarios.
3. **Advanced GPU Optimization and Model Quantization**
   Although the current version utilizes GPU for inference acceleration, future updates could include better memory management, model quantization (e.g., 4-bit/8-bit), or using more optimized backends (like TensorRT) to further reduce latency and boost performance on large-scale documents.
4. **Extended File Format Support**
   To improve versatility, the system can be enhanced to support additional file types such as DOCX, XLSX, HTML, and advanced scanned image formats. Enhanced OCR models (like LayoutLM or TrOCR) may also be used to improve extraction from complex or structured documents.
5. **Cross-Platform and Multi-Device Accessibility**
   Transforming the app into a full-featured web and mobile application with user authentication, session persistence, and cloud sync would greatly enhance usability across devices and for multiple users.
6. **Contextual Chat History and Feedback Mechanism**
   Incorporating session-based memory for chat history, real-time document answer highlighting, and user feedback mechanisms (e.g., rating or flagging answers) will provide a richer and more interactive user experience.
7. **Domain-Specific Fine-Tuning**
   Fine-tuning the LLaMA model with domain-specific datasets (e.g., legal, healthcare, academic) will improve the chatbot's accuracy and contextual understanding in professional environments, increasing its practical relevance.

10. CONCLUSION

This project demonstrates the development of a Local LLM-powered chatbot that ensures data privacy and operates entirely offline. By using Retrieval-Augmented Generation (RAG), the system retrieves relevant document content and generates context-aware responses with the local LLaMA model. The solution offers fast, accurate, and secure document-based question answering, making it ideal for environments requiring offline capability and privacy.

While the system performs effectively, future enhancements such as support for multiple languages, voice input, and GPU acceleration could further improve its functionality. Overall, this project provides a robust, privacy-preserving AI solution for document comprehension and querying.