

# **Design and Implementation of a Library AI Assistant: A RAG-Based Chatbot Leveraging Large Language Models (LLMs)**

**Author:** Mir Owahed Ali, Project Engineer (Content Curation), NDLI, IIT Kharagpur.

**Dr. B. Sutradhar, Librarian & Joint-PI, NDLI, Central Library, IIT Kharagpur.**

**Abstract** - This paper presents the design and implementation of a library assistant using Retrieval-Augmented Generation (RAG) architecture and Large Language Models (LLMs). The assistant enables users—including students, researchers, staff, and general readers—to ask natural language questions and receive accurate, brief responses. The system ingests content from publicly accessible user-facing documents, such as website text, user manuals, guides (PDF, PPT), and other informational resources.

It uses RAG pipeline, integrating embedding models such as nomic-embed-text and large language models (LLMs) like Mistral and LLaMA. Vector databases such as Supabase and FAISS are used to store and retrieve document embeddings. The assistant can be hosted in various environments—locally, in the cloud, or in hybrid setups—depending on performance and accessibility.

The paper discusses the implementation strategy, document pre-processing, retrieval logic, and chatbot response generation. The goal is to make library more accessible, avoiding the need to navigate pages or static FAQs. Some sample queries demonstrate how the AI assistant retrieves and synthesises answers effectively, making it a powerful tool for enhancing library services in the age of AI.

**Keywords:** Library AI Assistant; Retrieval-Augmented Generation (RAG); Large Language Models (LLMs); Semantic Search; Library Chatbot; Natural Language Processing (NLP)

## **1. Introduction**

Digital libraries have improved access to knowledge resources, but users often struggle to find specific information in unstructured content like help documents, guides, and service pages. Traditional search systems depend on keywords, which has the limitation for users who is unaware of exact terms.

Large Language Models (LLMs) can sometimes give incorrect or misleading answers if they are not given with content from a specific domain and up-to-date information. Retrieval-Augmented Generation (RAG) solves this issue by integrating a document retrieval step with LLMs, allowing the model to generate accurate and brief answer.

This paper presents both a conceptual framework and an implementation approach for building a Retrieval-Augmented Generation (RAG)-based AI assistant to enhance library services. The system aims to help users access library-related information—such as borrowing policies, access rules, and service usage through a natural language interface. Built using LLMs, vector databases, and local or cloud deployment, the assistant supports low-cost implementation.

## **2. Problem Statement**

Library users often struggle to find specific information quickly due to keyword-based search limitations and the need to manually browse multiple web pages and documents.

### 3. Objective

The objective of the RAG (Retrieval-Augmented Generation) AI Chatbot for libraries is to provide users with accurate, concise, and context-aware access to library information. By leveraging unstructured data such as library website content, guides, policies, and reference materials in formats like PDFs, Word documents, and PowerPoint slides, the chatbot aims to:

- **Enhance Information Discovery:** It help users quickly find relevant information, services, and library manuals through natural language queries.
- **Deliver Brief and Summarized Responses:** Provide concise answers to user queries by summarizing the most relevant information from multiple documents.
- **Simplify Information Access:** Allow users to obtain responses without relying on keyword-based search or navigating complex library websites.
- **Support Self-Service:** Reduce the workload on library staff by handling frequently asked questions about library services, borrowing policies etc.
- **Improve User Experience:** It makes accessing library knowledge more engaging and accessible to students, researchers, and faculty.
- **Save User Time:** Minimize the effort and time spent searching through multiple sources to find answers.

### 4. System Overview

AI assistant is designed to answer user queries by retrieving information from non-catalogue resources like PDFs, presentations, and web pages. It is open to all users—students, staff, researchers, and the public—without any access restrictions. The assistant supports a wide range of questions, including library user guide, available services.

### 5. Methodology

The AI chatbot follows broadly two steps **i) data ingestion ii) query retrieval**. In data ingestion, data sources such as web text, manuals and guides in various format are collected. Then data are parsed. It can be converted into LLM ready i.e., Markdown or JSON format and it is optional. The converted texts are split into smaller chunks. Then, chunks are embedded and converted into numerical representation i.e., vector format. Then vector data are stored into vector database. In query retrieval, user queries are embedded and searches into the vector databases using cosine similarity. It retrieves the contextual data and passes into the LLM. As per system prompt, LLM generates response. The architecture is discussed in the following.

#### Proposed Architecture

##### 5.1 Document Sources

The system is designed to ingest content from multiple sources related to library services and user support. These include:

- **Library Website Pages:** Library website pages are periodically crawled or scraped. The html data is then transformed into LLM ready, it may be Markdown or JSON format.
- **User-Facing Documents:** Static resources such as user manuals, announcements, help guides and training materials in various formats like PDF, PPT, DOCS) are collected. It is also converted into LLM ready.

## 5.2 RAG (Retrieval-Augmented Generation) Pipeline Design

The RAG pipeline is the core of the system to generate semantic responses.

- **Ingestion & Chunking:** The large documents are divided into smaller, meaningful chunks using tools like the Recursive Character Text Splitter. 1000 or 2000 character is set for chunking and character overlap can be 200. Every LLMs has a context window limit to process at once that's why chunking process is executed.
- **Embedding Generation:** Each chunk is converted into vector representation using embedding models. This can be done using open-source models like nomic-embed-text or proprietary APIs like OpenAI's text-embedding-3-small for high-quality embeddings. Embedding dimensions like 1536 is used to capture the semantic relation among words.
- **Vector Database:** The generated embeddings are stored in a vector space (e.g., FAISS, ChromaDB, or Supabase vector database). It helps to search and retrieve relevant content when a query is asked. To retrieve semantic data, cosine similarity is used.
- **Query Processing:** When a user asks a question, it is converted into a vector using the same embedding model which was used for document embedding. The system performs cosine similarity search in the vector database to retrieve the most relevant chunks. The top k (k=3 or 5) most similar chunks, along with the query, are sent to the LLM to generate response.

## 5.3 Language Model & Response Generation

The system uses lightweight, open-source large language models (LLMs) such as Meta Llama, IBM Granite, Mistral, OpenAI's gpt-oss or similar variants. These models are capable of processing natural language and generating responses. Model can be deployed in the following ways:

- **Local Deployment:** Models can be run locally using tools like Ollama, Docker Model Runner. It requires GPU or CPU based compute device.
- **Cloud LLM provider:** Cloud APIs like OpenRouter, Groq and HuggingFace offer access to fast, open-source LLMs without requiring local compute. This reduces infrastructure overhead. Free API key for the above services is provided with usage limit.

## 5.4 Chat Memory Integration

The system stores user inputs and assistant responses during a session using either Buffer Window Memory or PostgreSQL-based chat memory. For PostgreSQL-based db, the free Supabase cloud API can be used, though it has usage limitations.

## 5.5 RAG Engine / Agent with System Prompt

It is used to coordinates retrieval and response generation with System Prompt. It enhances response consistency. The following prompt can be used:

- You are a helpful library AI assistant.
- Your task is to answer user queries based only on the information retrieved from the vector database.
- Use the retrieved documents to provide factual, concise, and helpful answers.
- Avoid speculation or providing information not supported by the retrieved content.
- If the answer cannot be found in the retrieved knowledge, inform the user politely and guide them to contact the appropriate library section for support.

For example:

“I'm sorry, I couldn't find relevant information. Please contact the Library Services Team or <mailto:library-support@example.com> for accurate assistance.”

Always maintain a friendly, clear, and professional tone.

## 5.6 Framework and Its deployment

The backend application can be developed using Python and LangChain. It can also be developed using n8n, flowise or langflow. Backend platform can be hosted in the cloud. It ensures better performance and easier maintenance. It can also be deployed in the local compute which is more cost-effective.

## 5.7 UI of chatbot

The backend can be integrated with existing library website. The code snippet can be added in the html body tag of the library site. It shows the chat bubble on the library site. Or separate frontend can be developed using python streamlit or gradio framework.

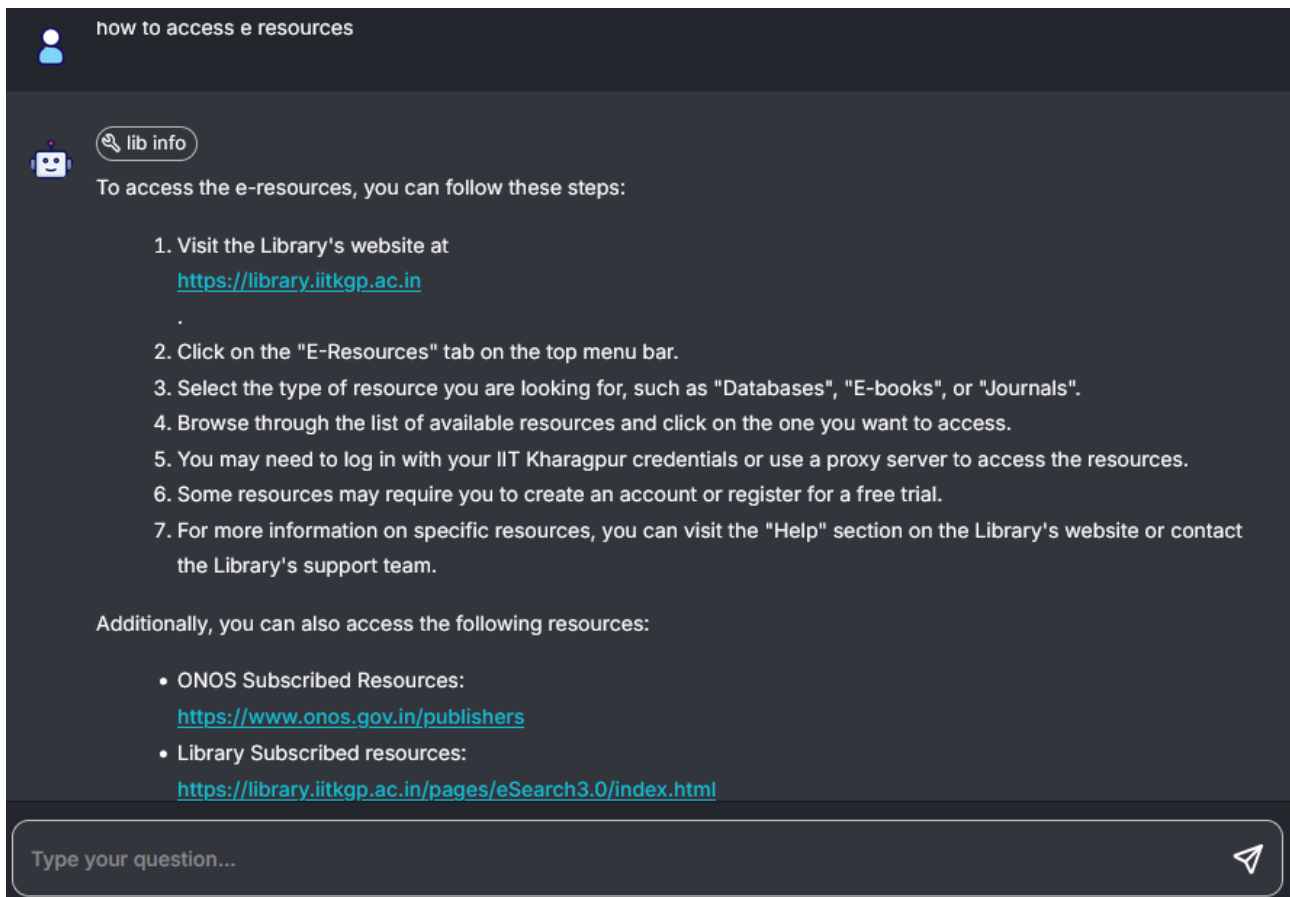


Fig.1. UI of Library AI Chat-bot

## 6. Data Collection

To implement the proposed AI chatbot, data were gathered from the Central Library website of IIT Kharagpur (<https://library.iitkgp.ac.in/>). The website content was collected using crawling and scraping.

In addition, library manuals, user guides, announcements, and training materials were compiled in multiple formats, including PDF, Word documents, and spreadsheets. These resources provide detailed information on services, borrowing policies and user instructions.

All collected data were then prepared for pre-processing to data ingestion process.

## 7. System Architecture Diagram

### Data Ingestion:

Data loading → Document parsing → Chunking → Embedding → Vector storage

### Query Retrieval:

User Query → Embedding → Vector DB Search → Context Retrieval → LLM + Prompt → Response

#### RAG Process- I

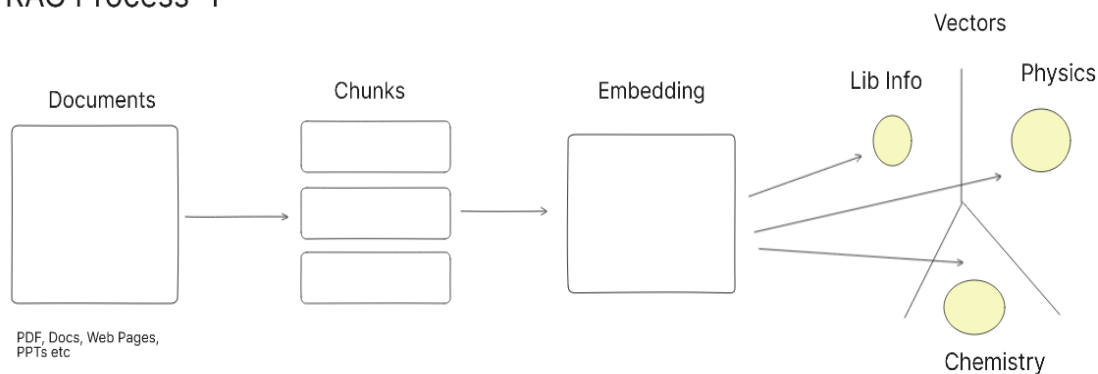


Fig.2. Data Ingestion diagram

#### RAG Process- II

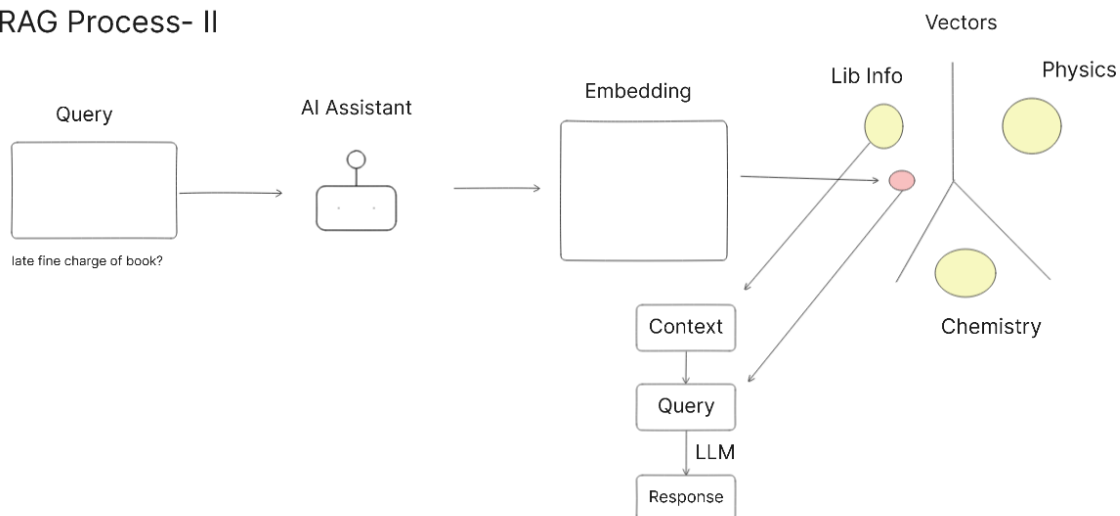


Fig.3. Query Retrieval diagram

## 8. Results and Discussions

**User:**

how to check my written paper originality

**AI Assistant:**

You can check the originality of your written paper using Turnitin, a plagiarism check tool and service available in the library. To do this, you need to upload your document to Turnitin and then send a request to the Digital Library section with the required details, such as Class ID, Author Name, Title of the Document, and Submission ID. The library will then verify the information and send you the AI report. For more information, you can contact the Digital Library section at: [console@library.iitkgp.ac.in](mailto:console@library.iitkgp.ac.in)  
[hbiswal@library.iitkgp.ac.in](mailto:hbiswal@library.iitkgp.ac.in)

**User:**

late fine charge of book

**AI Assistant:**

One rupee (Rs. 1/-) per day per book will be collected from the user as overdue charge.  
[/libsections.ht...](#)

**User:**

how to get sem question paper

**AI Assistant:**

You can access the semester-exam papers through the library's **Question-Paper portal**:

- **Web portal:** <http://10.18.24.75/peqp/> – the site lists the collected SEM exam papers for various courses.
- **If you need a specific paper that is not listed**, you may request it by emailing the console team at [console@library.iitkgp.ac.in](mailto:console@library.iitkgp.ac.in).

For any further assistance, contact the Library Services Team or mail [bsutra@library.iitkgp.ac.in](mailto:bsutra@library.iitkgp.ac.in).

The above responses shows that the AI chatbot performs well. The response generated by the LLM is retrieved from the RAG vector database. The term used in the user input is not mentioned in the source docs. It ensures the **semantic** responses. The chatbot interprets natural language queries and generates specific and summarize responses. It helps user to get specific url that provides more information.

## 9. Voice User Input and User Feedback Mechanism

User can interact with the AI assistant using voice by integrating speech-to-text models like OpenAI Whisper or Groq Whisper. This feature converts spoken language into text and enhances accessibility. It can be enabled a thumbs up/down feedback system to improve response quality and refine the assistant's performance.

## 10. Monitor and Analyse

To enhance the performance and usefulness of the AI assistant, monitoring tools like **LangSmith** is integrated using a free API key. LangSmith enables tracing and analysis of user queries and assistant responses in real time. By reviewing conversation logs, developers can identify gaps in response accuracy, refine system prompts, and improve retrieval quality. Analysing user input and frequently asked questions helps libraries to introduce new services. The following table shows a sample user queries, responses, error, time and latency of the chatbot.

Name	Input	Output	Error	Start Time	Latency
Chatbot	Mail id of the librarian	I'm happy to help! According to the retri...		8/7/2025, 3:40:44 PM	0.70s
Chatbot	Who is the librarian?	The librarian at Central Library, IIT Khar...		8/7/2025, 3:39:41 PM	2.52s
Chatbot	provide KEDLD url	The KEDLD (Knowledge and Education ...		8/7/2025, 3:34:54 PM	3.19s
Chatbot	how to submit paper in KEDLD	Based on the output of the tool call, it a...		8/7/2025, 3:33:15 PM	3.50s
Chatbot	provide kedld brochure		Cannot read pro...	8/7/2025, 3:25:28 PM	2.63s

Table 1 – Performance of Chat-bot

## 11. Challenges

- **Low-Quality Source Files:**  
Scanned or poorly formatted source files can lead to missing or unreadable text when processed. Always parse the data carefully. For best results, convert the data into an LLM-friendly format such as JSON or Markdown so the model can understand it better.
- **Infrastructure Costs:**  
Using CPU-based servers to run LLMs leads to higher latency and more resource consumption. Switching to GPU-based servers improves performance but it has higher costs.
- **Hallucination Risks:**  
Language models can sometimes give wrong or made-up answers, especially when the question is not clear.
- **LLM Selection:**  
The responses can vary depending on which Large Language Model (LLM) and version it uses. Choose the right LLM by reviewing and testing its responses multiple times to see which one works best for the chatbot.

## 12. Future Work & Improvement

- **Metadata Integration:**  
Integrate structured data such as library catalogue metadata (e.g., OPAC) or Institutional Digital Repository (IDR) metadata to retrieve library resources using semantic search.
- **Multilingual Query Support:**  
Enable users to interact in regional languages such as Bengali or Hindi, making the assistant more accessible to local library communities.
- **Service Optimization Through Insights:**  
Analyse user queries to identify information gaps, helping libraries enhance existing services or introduce new services based on user needs.

## 13. Conclusion

This work successfully demonstrates the design and implementation of a Retrieval-Augmented Generation (RAG)-based AI assistant to provide precise answers to user queries. The system integrates multiple key components—document parsing and chunking, embedding generation, vector database storage (FAISS, Supabase), and response generation using language models (LLMs) such as Mistral and LLaMA.

It shows flexible deployment strategy, supporting both local and cloud-based hosting. The inclusion of chat memory improves continuity of chat, while monitoring and feedback mechanisms help in refining system performance over time.

This implementation shows that RAG-powered chatbots can play a key role in creating the next generation of smart library systems.

## References:

- [1] Ilapaka, A., & Ghosh, R. (2025, May 23–24). *Comprehensive RAG-based LLM for AI-driven mental health chatbot*. In *Proceedings of the 2025 7th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (ICHORA)*. IEEE.  
<https://doi.org/10.1109/ICHORA65333.2025.11017017>
- [2] Saha, B., & Saha, U. (2024). *Enhancing international graduate student experience through AI-driven support systems: A LLM and RAG-based approach*. In *Proceedings of the 2024 International Conference on Data Science and Its Applications (ICoDSA)* (pp. 300–304). IEEE.  
<https://doi.org/10.1109/ICoDSA62899.2024.10651944>
- [3] Suryavanshi, K., Thikekar, N., Pawar, R., & Ashtekar, S. (2025). *Implementation of RAG based question-answering application*. In *Proceedings of the 2025 International Conference on Data Science and Business Systems (ICDSBS)* (pp. 1–6). IEEE.  
<https://doi.org/10.1109/ICDSBS63635.2025.11031968>
- [4] Comia, L. V. (2025). *Chatbot for student discipline handbook-related queries: A RAG-based LLM using Llama-3 approach*. In *2025 11th International Conference on Web Research (ICWR)* (pp. 306–312). IEEE. <https://doi.org/10.1109/ICWR65219.2025.11006165>

- [5] <https://ollama.com/>
- [6] <https://docs.docker.com/ai/model-runner/>
- [7] <https://docs.smith.langchain.com/>
- [8] <https://openrouter.ai/>
- [9] <https://flowiseai.com/>
- [10] <https://www.langflow.org/>
- [11] <https://groq.com/>
- [12] <https://huggingface.co/>
- [13] <https://supabase.com/docs>
- [14] <https://library.iitkgp.ac.in/>