End-to-End Sanskrit Retrieval-Augmented Generation (RAG) Chatbot

AI/ML Internship Project Report

1. Abstract

This project presents the development of an end-to-end Retrieval-Augmented Generation (RAG) chatbot designed to answer questions in Sanskrit. The system integrates hybrid information retrieval (semantic + keyword search) with a Large Language Model (LLM) to generate context-grounded answers. It supports both Devanagari script and Latin transliteration input, making it accessible to a wider audience. The architecture is modular, scalable, and optimized to reduce hallucinations while improving answer relevance.

2. Introduction

Large Language Models (LLMs) are powerful but often generate hallucinated or ungrounded responses. To overcome this, Retrieval-Augmented Generation (RAG) systems combine information retrieval with language generation, ensuring that responses are based on verified documents.

This project builds a Sanskrit Question Answering System using a hybrid retrieval approach and an LLM, deployed through a FastAPI backend with a web interface.

3. Problem Statement

Traditional LLMs:

- Do not have access to external documents at inference time
- May produce incorrect or hallucinated answers
- Cannot directly support structured Sanskrit document QA

Objective:
Build an **end-to-end RAG chatbot** that:

- Accepts Sanskrit or transliterated queries
- Retrieves relevant Sanskrit text passages
- Generates grounded answers
- Minimizes hallucinations

4. System Architecture Overview

The system consists of two major pipelines:

Offline Pipeline (Ingestion & Indexing) refer Fig. 1

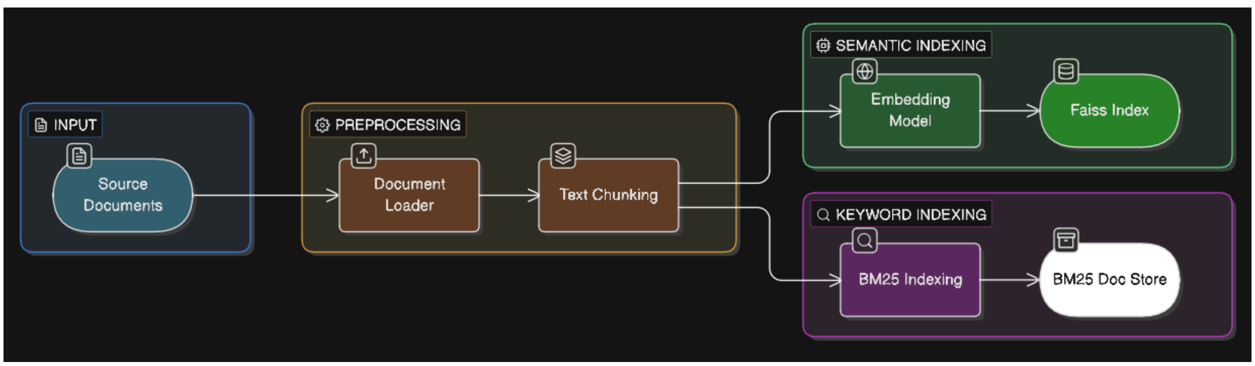
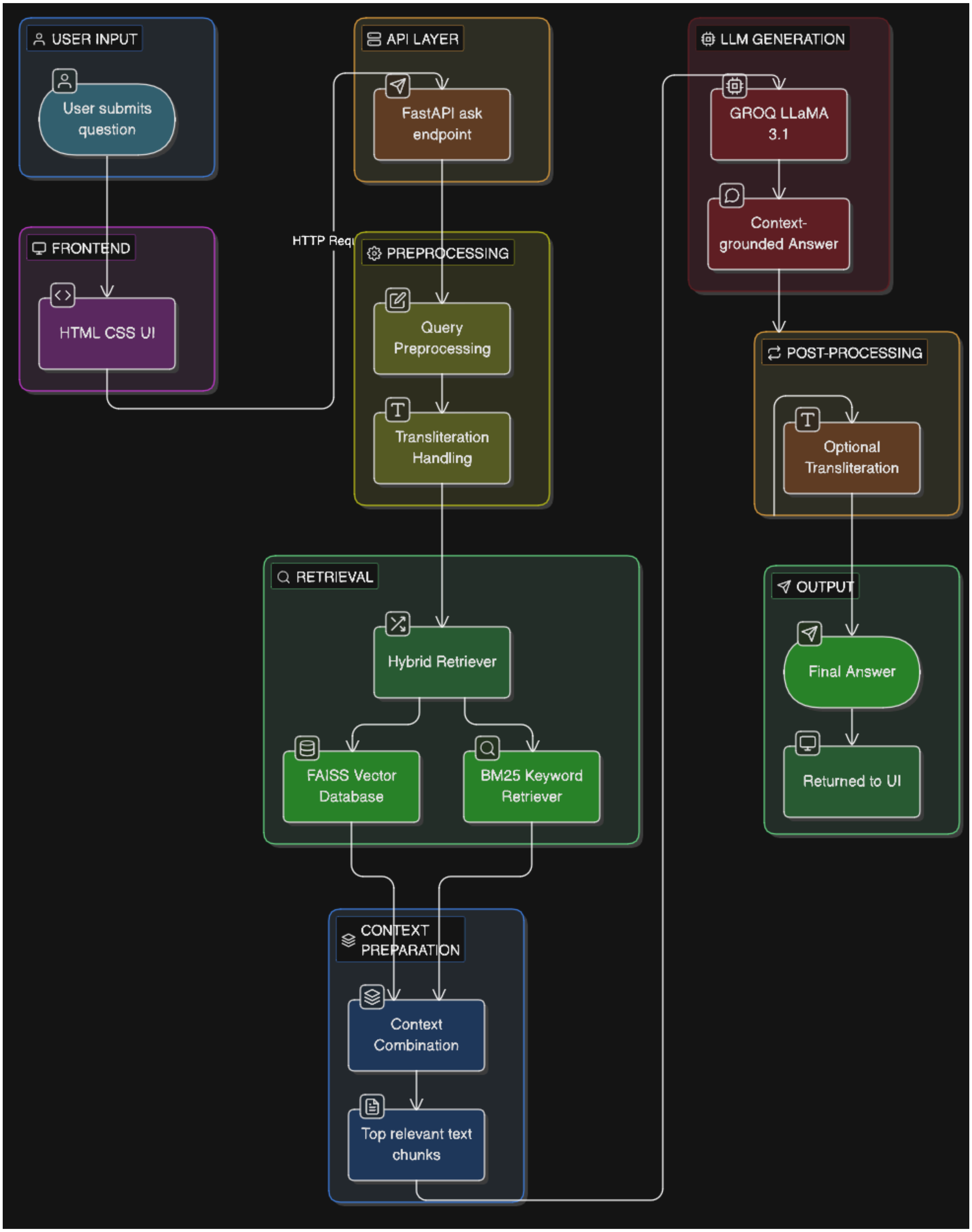


Fig. 1

Fig . 1 - Used to prepare the knowledge base.

Online Pipeline (Question Answering)

Used to answer user queries in real time, Refer Fig. 2



USER INPUT
User submits question
FRONTEND
HTML CSS UI
HTTP Requ
API LAYER
FastAPI ask endpoint
PREPROCESSING
Query Preprocessing
Transliteration Handling
RETRIEVAL
Hybrid Retriever
FAISS Vector Database
BM25 Keyword Retriever
CONTEXT PREPARATION
Context Combination
Top relevant text chunks
LLM GENERATION
GROQ LLaMA 3.1
Context-grounded Answer
POST-PROCESSING
Optional Transliteration
OUTPUT
Final Answer
Returned to UI

Fig. 2

5. Offline Pipeline – Knowledge Base Creation

This stage runs once before deployment.

Step 1: Document Loading

Supports:

- `.txt`
- `.docx`
- `.pdf`

Sanskrit documents are loaded and converted into plain text.

Step 2: Text Chunking

Documents are split using:

- **Chunk size:** 800 characters
- **Chunk overlap:** 150 characters

This ensures semantic continuity between chunks and improves retrieval quality.

Step 3: Embedding Generation

Each chunk is converted into a vector using:

Model: `intfloat/multilingual-e5-base`

These embeddings capture semantic meaning for similarity search.

Step 4: Vector Index (FAISS)

All chunk embeddings are stored in a **FAISS vector database** for fast semantic retrieval.

Step 5: Keyword Index (BM25)

Chunks are also indexed using **BM25**, a keyword-based retrieval method. This complements vector search by matching exact terms.

Offline Output

Component	Purpose
FAISS Index	Semantic search
BM25 Index	Keyword search

6. Online Pipeline – Question Answering

This pipeline runs every time a user asks a question.

Step 1: User Input

User enters a question in:

- Sanskrit (Devanagari)
  OR
- Latin transliteration (e.g., *ghantakarNasya katha kathaya*)

Step 2: Transliteration Detection

If Latin script is detected:

- It is converted into Devanagari before retrieval

Step 3: Hybrid Retrieval

Two retrieval methods run in parallel:

Retriever	Function
FAISS (Vector Search)	Finds semantically similar text
BM25 (Keyword Search)	Finds exact word matches

Results are combined using a **weighted ensemble**:

Hybrid Score = 0.6 × FAISS + 0.4 × BM25

This improves both:

- Recall (finding relevant passages)
- Precision (reducing noise)

Step 4: Context Construction

Top relevant chunks are merged to form a **context block** that is passed to the LLM.

Step 5: LLM-Based Answer Generation

Model: LLaMA 3.1 (via Groq API)

Prompt rules:

- Use only the provided context
- Do not add external knowledge
- Answer only in Sanskrit
- If not found → reply “न ज्ञायते”

Step 6: Output Formatting

If the user used transliteration:

- The Sanskrit answer is converted back to Latin script

7. Technology Stack

Layer	Technology
Backend API	FastAPI
Frontend	HTML, CSS
LLM	LLaMA 3.1 (Groq)
Embeddings	multilingual-e5-base
Vector Database	FAISS
Keyword Search	BM25
Text Chunking	RecursiveCharacterTextSplitter
Evaluation	LLM-as-Judge + (F1/EM metrics)has to be done

8. Evaluation Methodology

The system was evaluated using two approaches:

1. LLM-as-Judge Evaluation

Each response was evaluated on:

- Correctness
- Groundedness

- Hallucination detection

(Has to be done): Automated Metrics

- **Exact Match (EM)** – Exact match with ground truth
- **F1 Score** – Partial word overlap
- **Context Recall** – Whether retrieval fetched correct supporting tex

9. Results:

1.1 User Interface

Fig 3 : Web interface of the Sanskrit RAG Assistant where users can enter queries.

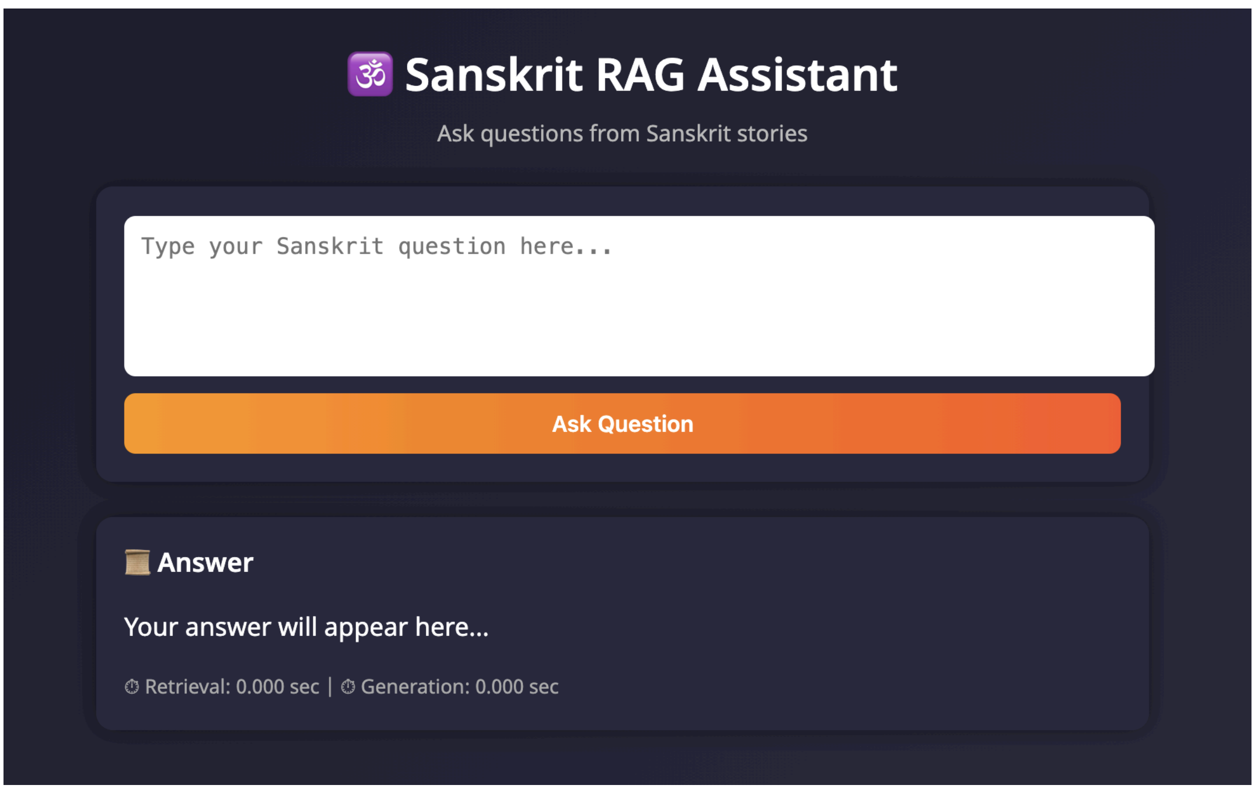


Fig. 3

Fig 4,5,6: System answering a Sanskrit query using hybrid retrieval and LLM grounding.

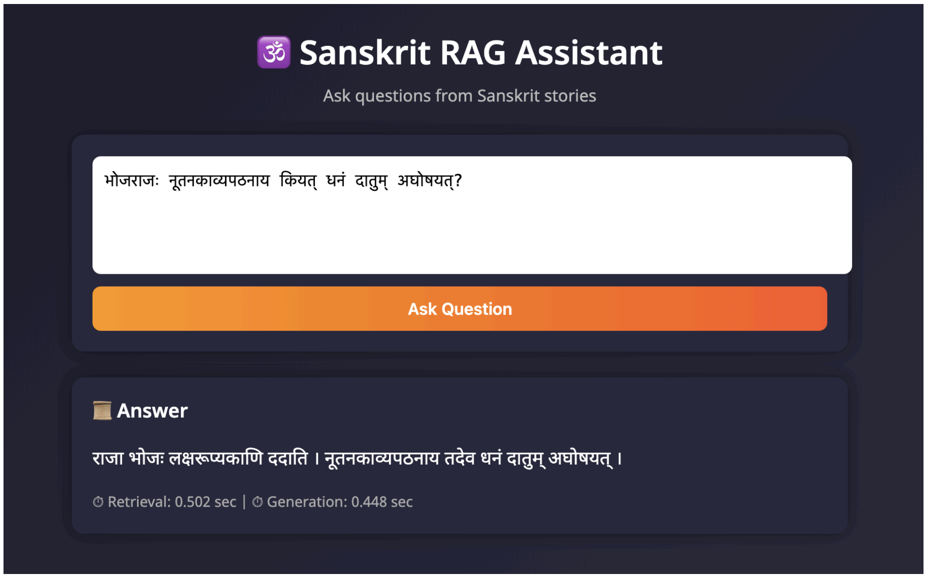


Fig. 4

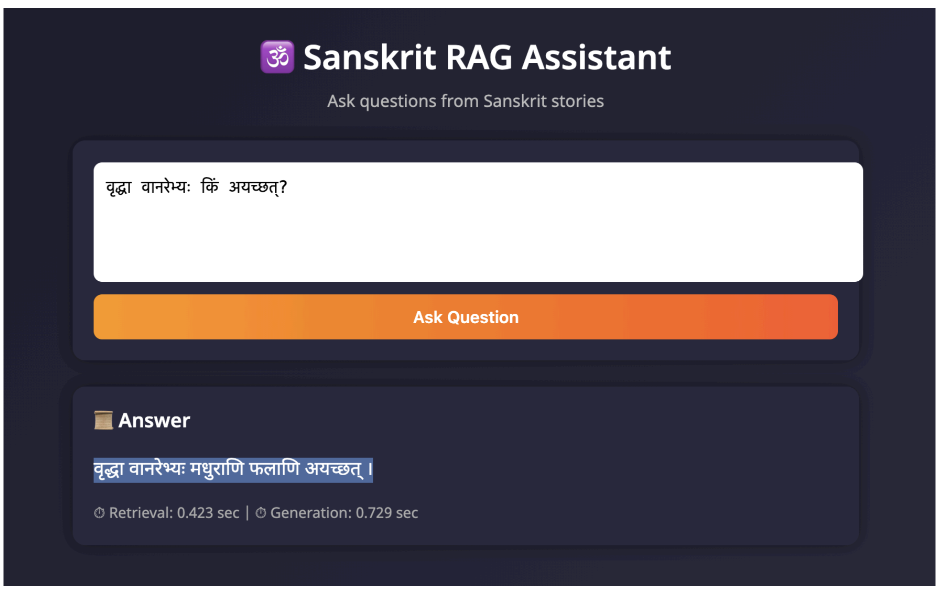


Fig. 5

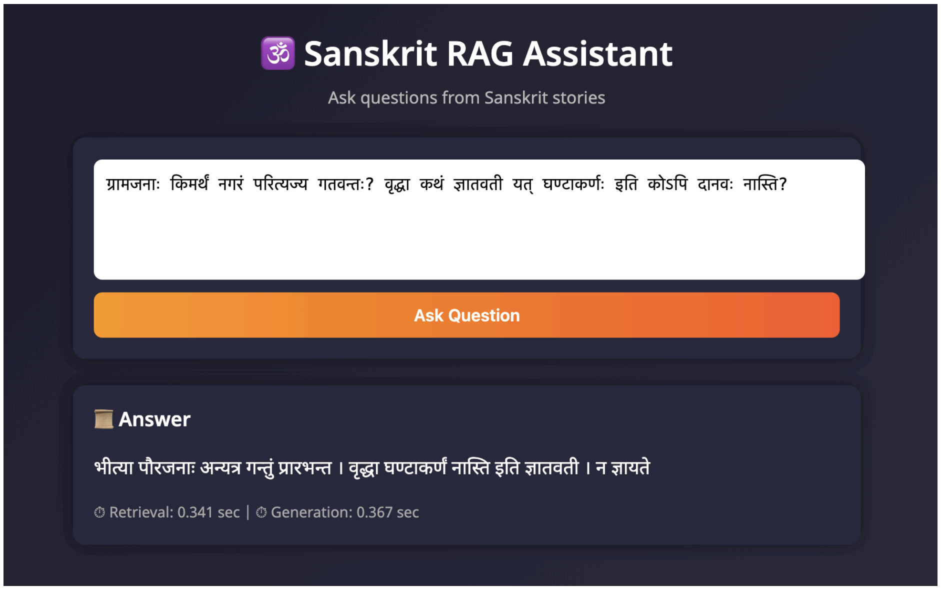
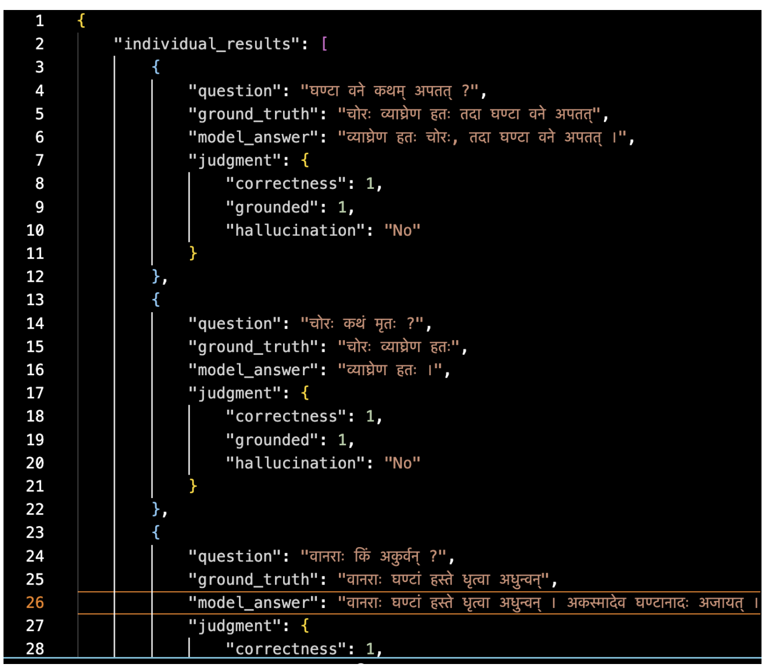


Fig. 6

1.2 Performance Analysis.

Fig 7.sample of evaluated data

Fig 8. Overall Metrics



```
{
    "individual_results": [
        {
            "question": "घण्टा वने कथम् अपतत् ?",
            "ground_truth": "चोरः व्याघ्रेण हतः तदा घण्टा वने अपतत्",
            "model_answer": "व्याघ्रेण हतः चोरः, तदा घण्टा वने अपतत् ।",
            "judgment": {
                "correctness": 1,
                "grounded": 1,
                "hallucination": "No"
            }
        },
        {
            "question": "चोरः कथं मृतः ?",
            "ground_truth": "चोरः व्याघ्रेण हतः",
            "model_answer": "व्याघ्रेण हतः ।",
            "judgment": {
                "correctness": 1,
                "grounded": 1,
                "hallucination": "No"
            }
        },
        {
            "question": "वानराः किं अकुर्वन् ?",
            "ground_truth": "वानराः घण्टां हस्ते धृत्वा अधुन्वन्",
            "model_answer": "वानराः घण्टां हस्ते धृत्वा अधुन्वन् । अकस्मादेव घण्टानादः अजायत् ।
            "judgment": {
                "correctness": 1,
```


Fig. 7

```
"overall_metrics": {
    "correctness_accuracy": 0.7,
    "grounded_rate": 0.6,
    "hallucination_rate": 0.3
}
```

Fig. 8

10. Key Features

✔ Hybrid Retrieval (FAISS + BM25)
✔ Sanskrit & Transliteration Support
✔ Context-Grounded Answers
✔ Modular Code Architecture
✔ FastAPI Deployment
✔ Evaluation Framework

11. Challenges and Solutions

Challenge	Solution
Handling transliteration	Script detection + conversion module
Hallucinated answers	Strict LLM prompting + retrieval grounding
Poor retrieval	Hybrid FAISS + BM25 search
Response latency	Groq high-speed inference

12. Conclusion

This project demonstrates a fully functional end-to-end RAG chatbot tailored for Sanskrit question answering. By integrating hybrid retrieval techniques with LLM-based generation, the system produces accurate, grounded, and context-aware answers while minimizing hallucinations. The modular design ensures scalability and adaptability for future enhancements.