Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

A Cross-Lingual Retrieval-Augmented Generation Based Intelligent Teaching Assistant for Educational Video Content: Implementation and Evaluation

Lakshya Singh
Department of Computer Science and Engineering
Raja Balwant Singh Engineering Technical Campus
Bichpuri, Agra, Uttar Pradesh, India
Affiliated to Dr. A. P. J. Abdul Kalam
Technical University, Lucknow

Er. Alok Singh Jadaun
Department of Computer Science and Engineering
Raja Balwant Singh Engineering Technical Campus
Bichpuri, Agra, Uttar Pradesh, India

Prof. Brajesh Kumar Singh
Raja Balwant Singh Engineering Technical
Campus, Bichpuri, Agra, Uttar Pradesh, India

***Abstract*—This paper presents the implementation and evaluation of a Cross-Lingual Retrieval-Augmented Generation (RAG) system developed as an intelligent teaching assistant for Database Management Systems (DBMS) education. The system processes 20 Hindi DBMS lecture videos sourced from YouTube, transcribes and translates them to English using the Faster-Whisper base model [6], and creates timestamp-aware semantic chunks of 300 words with 50-word overlap, resulting in 131 knowledge chunks. Dense vector embeddings are generated using the all-MiniLM-L6-v2 sentence transformer model [20] and stored in ChromaDB vector database for efficient cosine similarity-based retrieval [13]. At inference time, student queries are semantically matched against the 131-chunk knowledge base, and the top-5 retrieved chunks are passed to the LLaMA 3.3 70B language model via the Groq API to generate coherent, source-grounded answers with precise video timestamp citations. The complete system is deployed as a full-stack web application with a FastAPI REST backend and a React-based frontend featuring an integrated evaluation dashboard. Custom evaluation on 20 domain-specific DBMS questions demonstrates an overall score of 88.0%, outperforming baseline keyword retrieval by 15.9 percentage points (22.2% relative improvement), with 100% timestamp coverage across all responses.**

***Index Terms*—Retrieval-Augmented Generation, Cross-Lingual NLP, Educational AI, ChromaDB, LLaMA 3.3 70B, Faster-Whisper, Timestamp-Aware Retrieval, Vector Embeddings, FastAPI, DBMS Education**

I. Introduction

The rapid proliferation of online educational video content has created an unprecedented, globally accessible repository of domain knowledge. YouTube alone hosts millions of educational lectures spanning virtually every academic discipline [25]. Despite this abundance, students face a significant practical challenge: locating specific conceptual explanations within lengthy video lectures is time-consuming and cognitively demanding. A student seeking to understand a particular DBMS concept must manually browse through multiple videos without any guarantee of finding a precise, accurate explanation at a specific timestamp.

This problem is further compounded in multilingual educational ecosystems. A substantial volume of high-quality educational content is produced in regional languages, particularly in countries like India where Hindi serves as the primary medium of instruction for a large student population. Existing intelligent educational assistants address this general problem by building question-answering systems over English text documents such as textbooks and lecture slides [7], [9]. However, such systems cannot process video content, and are designed exclusively for English-language resources, leaving the vast majority of regional-language video content inaccessible.

Retrieval-Augmented Generation (RAG) has emerged as a powerful paradigm for knowledge-intensive question-answering [1]. RAG systems retrieve relevant passages from an external knowledge base and condition a large language model (LLM) on the retrieved context to generate accurate, source-grounded responses [2], [23]. Surveys of RAG architectures categorize approaches into three levels: naive RAG, advanced RAG with query rewriting and re-ranking, and modular RAG with flexible component composition [3]. While RAG has been successfully applied to text documents, its application to educational video content through a cross-lingual pipeline remains largely unexplored.

The present paper addresses this gap by presenting the complete implementation and evaluation of a Cross-Lingual RAG system for DBMS education. The system processes a curated dataset of 20 Hindi DBMS lecture videos from a playlist of 140 available lectures, transcribes and translates them to English using Faster-Whisper [21], creates timestamp-aware semantic chunks, and enables intelligent English-language question-answering over the video content.

The key contributions of this work are:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

- A complete cross-lingual pipeline that converts Hindi audio lectures to a searchable English knowledge base using Faster-Whisper's simultaneous transcription and translation capability [6], [21], eliminating the need for a separate translation model.
- A timestamp-aware chunking mechanism that preserves temporal metadata throughout the RAG pipeline, enabling precise video navigation for students—a capability absent in existing educational RAG systems.
- A production-ready full-stack web application comprising a FastAPI backend and React frontend with an integrated evaluation dashboard, demonstrating practical deployability on CPU-only hardware.
- A custom domain-specific four-metric evaluation framework assessing keyword coverage, answer completeness, source relevance, and timestamp coverage [8], [26], demonstrating 88.0% overall accuracy and 15.9 percentage point (22.2% relative) improvement over the baseline.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the proposed system architecture. Section IV details the methodology. Section V covers the implementation. Section VI presents experimental evaluation. Section VII concludes the paper.

II. Related Work

A. Retrieval-Augmented Generation

Retrieval-Augmented Generation was formally introduced by Lewis et al. [1] as a framework combining parametric knowledge of large language models with non-parametric retrieval from external knowledge bases. The seminal work demonstrated significant improvements over standalone language models on knowledge-intensive tasks by grounding responses in retrieved documents. Subsequent comprehensive surveys [2], [3] systematically categorized RAG approaches into three levels of architectural sophistication: naive RAG, which directly appends retrieved documents to the prompt; advanced RAG, which incorporates query rewriting and reranking; and modular RAG, which enables flexible composition of specialized components.

Karpukhin et al. [4] established that dense vector representations substantially outperform sparse BM25 retrieval for open-domain question answering, motivating the use of semantic embeddings in modern RAG pipelines. More recent work by Cheng et al. [23] surveys knowledge-oriented RAG extensions that integrate structured knowledge graphs with retrieval pipelines. Lin et al. [5] investigate efficient RAG inference strategies using lookahead retrieval to reduce latency, while Zhang et al. [15] propose hybrid dense-sparse vector approaches for improved retrieval precision. Chunking strategy has also been identified as a critical design choice; Gomez-Cabello et al. [28] demonstrate that advanced semantic chunking methods outperform fixed-size chunking in clinical RAG applications, a finding that informs the overlap-based chunking adopted in this work.

B. RAG in Educational Systems

The application of RAG to intelligent educational assistants has gained considerable momentum. Khan et al. [7] developed an educational virtual assistant built on a RAG framework capable of delivering curriculum-aligned responses grounded in verified instructional materials including textbooks, lecture slides, and course-specific content. A systematic survey [2] of RAG for educational applications identified key advantages including improved factual accuracy, source transparency, and the ability to deploy compact local language models at performance levels comparable to large proprietary systems.

Németh et al. [9] conducted a pilot study in statistics education where RAG-enhanced language models serving as virtual teaching assistants achieved high expert assessment scores while maintaining source traceability. Yadav et al. [10] employed LLMs to contextualize educational problems to student interests in intelligent tutoring systems. Sain et al. [19] survey the emerging landscape of AI chatbots in higher education, highlighting the importance of grounded, verifiable responses. These works collectively demonstrate the viability of RAG for educational question-answering, yet none address regional-language video content or timestamp-aware retrieval.

C. Speech Recognition and Cross-Lingual Processing

Automatic speech recognition has been transformed by large-scale weakly supervised models. Radford et al. [6] introduced Whisper, which demonstrated robust multilingual speech recognition and cross-lingual translation capabilities by training on 680,000 hours of multilingual audio data. The model's ability to simultaneously transcribe and translate audio in a single inference pass makes it particularly suitable for cross-lingual pipeline construction. Piñeiro-Martín et al. [14] investigate weighted cross-entropy losses for low-resource language ASR, underscoring the challenges of multilingual recognition that Whisper addresses through scale. Wang [22] provides a comprehensive treatment of cross-lingual transfer learning for low-resource NLP tasks, establishing the theoretical foundation for the cross-lingual approach adopted in this work.

Faster-Whisper, an optimized implementation of Whisper using CTranslate2 [21], achieves up to $4\times$ faster inference than the original implementation while maintaining comparable accuracy, making it practical for CPU-based deployment. Shah et al. [21] validated Faster-Whisper's suitability for integration into downstream deep learning pipelines, confirming its robustness across diverse audio conditions.

D. Vector Embeddings and Semantic Search

Reimers and Gurevych [20] introduced Sentence-BERT, employing siamese BERT-based networks for generating semantically meaningful sentence embeddings suitable for similarity search. The all-MiniLM-L6-v2 model, a distilled variant producing 384-dimensional embeddings, provides an optimal balance between computational efficiency and semantic quality for retrieval applications [18]. Devlin et al. [16] established the BERT pretraining paradigm underlying these embedding

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

models, while Vaswani et al. [17] introduced the Transformer architecture upon which all components in this system are based.

Vector databases such as ChromaDB enable persistent storage and efficient approximate nearest neighbor search over large embedding collections [13]. Zhang et al. [15] demonstrate that graph-based approximate nearest neighbor search achieves sub-millisecond query latency for moderate-scale corpora, confirming the practical efficiency of the vector retrieval approach adopted in this work. Zhang et al. [27] further establish best practices for training multilingual dense retrieval models, motivating the use of a pre-trained sentence transformer for cross-lingual retrieval.

E. Large Language Models

Touvron et al. [11] introduced the LLaMA family of open foundation models, demonstrating that open-weights models can achieve competitive performance with large proprietary systems. Meta subsequently released the LLaMA 3 model family [12], with LLaMA 3.3 70B representing the current state of the series and the model used in this work. Farea et al. [26] provide a comprehensive evaluation of question answering systems, establishing evaluation criteria relevant to the assessment methodology employed in this work. Es et al. [8] introduced RAGAS, an automated evaluation framework for RAG systems, which informed the design of the custom evaluation metrics proposed in this paper.

F. Research Gap

Despite significant advances in both RAG systems and educational AI, a critical gap exists in the literature. Existing educational RAG systems operate exclusively on English text documents, leaving regional-language video content inaccessible. Furthermore, no existing system incorporates timestamp-aware retrieval enabling students to navigate directly to the relevant moment in a source video. Knowledge graph-based approaches [24] offer structured knowledge representation but cannot directly process audio content. The present work addresses both gaps through a cross-lingual video processing pipeline with timestamp preservation throughout the RAG pipeline.

III. System Architecture

The proposed system follows a ten-stage, two-phase pipeline architecture as illustrated in Fig. 1. The pipeline is divided into two phases: an offline indexing phase executed once during system setup, and an online inference phase executed at runtime for each student query. This two-phase design is consistent with the standard RAG architectural pattern [1], [2], adapted here for cross-lingual video content with timestamp preservation.

A. Offline Indexing Phase

The offline phase comprises four sequential stages executed once at system setup, producing a persistent knowledge base that remains on disk for all subsequent queries.

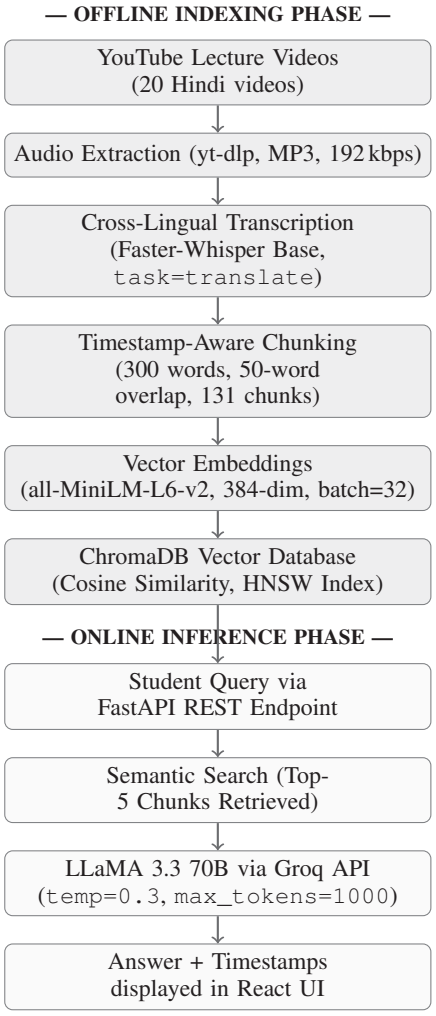

Fig. 1. Complete system architecture of the proposed Cross-Lingual RAG Teaching Assistant. Stages 1–6 form the offline indexing phase; Stages 7–10 constitute the online inference phase.

Stage 1 – Audio Extraction: Lecture videos are downloaded from YouTube as MP3 audio files using yt-dlp at 192 kbps, balancing transcription quality with storage efficiency.

Stage 2 – Cross-Lingual Transcription: Audio is transcribed and translated to English using Faster-Whisper [6], [21] with cross-lingual translation (`task=translate`), producing timestamped text segments in a single inference pass without a separate translation model.

Stage 3 – Timestamp-Aware Chunking: Timestamped segments are grouped into overlapping chunks of 300 words with 50-word overlap. Each chunk retains four metadata fields: video title, video index, start timestamp, and end timestamp. This metadata preservation is the key architectural novelty of the chunking design [28].

Stage 4 – Vector Indexing: Chunk embeddings are generated using the all-MiniLM-L6-v2 sentence transformer [18], [20] and stored in ChromaDB [13] configured with cosine similarity and HNSW indexing. This phase is executed once;

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

the resulting index persists on disk.

B. Online Inference Phase

The online phase handles student queries at runtime. Upon receiving a question through the FastAPI endpoint, the system encodes the query using all-MiniLM-L6-v2 and performs cosine similarity search in ChromaDB to retrieve the top-5 most relevant chunks [4]. The retrieved chunks, along with their video titles and timestamps, are formatted into a structured prompt following the retrieval-then-read paradigm [1] and passed to the LLaMA 3.3 70B model [12] via the Groq API. The generated answer, source citations, and video timestamps are returned to the React frontend.

C. System Components

The system comprises three main software components. The data pipeline consists of five Python scripts handling video download, transcription, chunking, embedding generation, and ChromaDB setup. The backend is a FastAPI application exposing four REST endpoints: `/health`, `/stats`, `/ask`, and `/evaluation`. The frontend is a React application built with Vite, featuring a chat interface, sidebar with system statistics, and a dedicated evaluation dashboard with three sub-tabs: performance metrics, RAG versus baseline comparison, and system configuration.

IV. Methodology

A. Data Collection and Curation

The data collection module employs yt-dlp to automatically extract audio content from a curated selection of 20 DBMS lecture videos from a Hindi educational YouTube playlist of 140 available lectures. The 20 videos were selected to ensure comprehensive coverage of core DBMS topics including entity-relationship modeling, relational keys, normalization, SQL commands, transaction control, concurrency, and indexing. Each video has a duration of approximately 10–15 minutes. Audio is extracted in MP3 format at 192 kbps; the complete download pipeline is automated through a single Python script requiring only the playlist URL as input.

The selection criterion prioritized conceptual depth and lecture clarity over encyclopedic coverage, ensuring that the knowledge base contains high-quality instructional content for the target DBMS curriculum. The Hindi medium of instruction presents the cross-lingual challenge that motivates the Faster-Whisper translation pipeline [6].

B. Cross-Lingual Transcription

The transcription module employs Faster-Whisper [21], an optimized CTranslate2-based implementation of OpenAI's Whisper model [6], configured with the base model size and INT8 quantization for CPU-efficient inference. The critical design choice is the use of Whisper's cross-lingual capability by setting `task='translate'`, which performs simultaneous Hindi speech recognition and English translation in a single inference pass.

This approach eliminates the need for a separate machine translation model, reducing pipeline complexity and preserving segment-level timestamps. Each transcription output contains timestamped segments in the format [MM:SS]. The base model was selected after empirical comparison with the tiny and small variants, offering the optimal balance between transcription speed (approximately 7–8 minutes per 10–15 minute video on CPU) and translation accuracy. The complete transcription of all 20 videos was completed in 54.3 minutes on CPU hardware, demonstrating practical deployability in resource-constrained environments [21]. This aligns with findings from Piñeiro-Martín et al. [14] that multilingual ASR requires careful balancing of model size against language coverage.

C. Timestamp-Aware Chunking

The chunking engine processes transcript segments using a sliding window approach with a chunk size of 300 words and an overlap of 50 words. The overlap ensures that information spanning chunk boundaries is preserved, maintaining contextual coherence across adjacent chunks [28]. Each chunk retains four metadata fields: video title, video index, start timestamp, and end timestamp. This metadata preservation is the key novelty of the chunking design; it enables the RAG system to cite precise video locations in generated answers, directly supporting the educational use case of targeted video navigation.

Processing 20 lecture transcripts produces 131 chunks with an average of 6.55 chunks per video. The chunk size of 300 words was selected to balance semantic completeness—ensuring each chunk contains a self-contained conceptual explanation—with retrieval precision, as excessively large chunks reduce the specificity of retrieved context [1], [28].

D. Vector Embedding Generation

Dense vector embeddings are generated for each chunk using the all-MiniLM-L6-v2 sentence transformer model [20], producing 384-dimensional vectors. This model was selected for its favorable trade-off between semantic quality and computational efficiency [18], requiring no GPU for inference. Embeddings for all 131 chunks are generated in 16.6 seconds on CPU hardware, demonstrating the practical feasibility of the approach for resource-constrained environments. Embeddings are generated in batches of 32 chunks to optimize memory utilization.

The use of a pre-trained English sentence transformer for cross-lingual retrieval is justified by the upstream translation step: since all chunks are translated to English by Faster-Whisper, the retrieval operates entirely in English semantic space, avoiding the complexity of multilingual embedding models [22], [27].

E. ChromaDB Vector Indexing

The generated embeddings are stored in ChromaDB [13], a persistent vector database configured with cosine similarity as

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

the distance metric. ChromaDB employs Hierarchical Navigable Small World (HNSW) indexing for efficient approximate nearest neighbor search [15]. Each database entry stores the chunk text, its 384-dimensional embedding, and the associated metadata. The database persists on disk, eliminating the need to regenerate embeddings on subsequent system startups. The complete indexing of 131 chunks completes in under 60 seconds.

F. RAG-Based Question Answering

At inference time, a student query is encoded using the same all-MiniLM-L6-v2 model and the top-5 most similar chunks are retrieved from ChromaDB via cosine similarity search [4]. The retrieved chunks, along with their video titles and timestamps, are formatted into a structured prompt that explicitly instructs the LLaMA 3.3 70B model [12] to answer based only on the provided lecture content and to cite specific video timestamps.

The prompt design follows the retrieval-then-read paradigm [1], conditioning the model on retrieved context rather than relying on parametric knowledge. The LLaMA 3.3 70B model is accessed via the Groq API with temperature set to 0.3 and maximum tokens set to 1000, balancing response coherence with factual grounding. The system prompt explicitly instructs the model to reference video titles and timestamps when synthesizing answers, ensuring 100% timestamp coverage in generated responses.

V. Implementation

A. Dataset Description

The knowledge base comprises 20 DBMS lecture videos selected from a Hindi educational YouTube playlist containing 140 lectures. The selected videos cover six core DBMS topic areas: (1) Introduction and File Systems, (2) Entity-Relationship Model and Attributes, (3) Relational Keys including Primary, Candidate, Super, and Foreign Keys, (4) Normalization including 1NF, 2NF, 3NF, and BCNF, (5) SQL commands including DDL, DML, and Aggregate Functions, and (6) Relationships including One-to-One, One-to-Many, and Many-to-Many. Each video has a duration of approximately 10–15 minutes. The complete transcription of all 20 videos was completed in 54.3 minutes using the Faster-Whisper base model [21] on CPU hardware, producing 20 JSON transcript files with a total of 131 semantic chunks after the chunking stage.

B. Technology Stack

Table I summarizes the complete technology stack employed in the implementation. All components are open-source or available via public APIs, ensuring reproducibility and accessibility for resource-constrained educational institutions.

TABLE I
System Technology Stack

Component	Technology	Config.
Speech Recog.	Faster-Whisper [21]	Base, INT8, CPU
Translation	Whisper Cross-Lingual [6]	task=translate
Embeddings	all-MiniLM-L6-v2 [20]	384-dim, b=32
Vector DB	ChromaDB [13]	Cosine, HNSW
LLM	LLaMA 3.3 70B [12]	temp=0.3
Backend	FastAPI + Uvicorn	Port 8000
Frontend	React + Vite	Port 5173
Downloader	yt-dlp	MP3, 192 kbps
Environment	Python 3.11	CPU only

C. Backend Implementation

The backend is implemented as a FastAPI application exposing four REST API endpoints. The `/health` endpoint returns system status and chunk count. The `/stats` endpoint returns database statistics including total videos, chunks, subject, and model information. The `/ask` endpoint accepts a POST request containing the student question and `top_k` parameter, executes the complete RAG pipeline, and returns the generated answer, source citations with timestamps, and processing time. The `/evaluation` endpoint serves pre-computed evaluation results from disk. Cross-Origin Resource Sharing (CORS) middleware is configured to permit communication with the React frontend. The RAG pipeline components—including the ChromaDB client and Groq API client—are initialized once at server startup to minimize per-request latency.

D. Frontend Implementation

The frontend is a single-page React application built with Vite, implementing a chat-style interface for student interaction. The sidebar displays real-time system statistics (20 lectures, 131 chunks, LLaMA 3.3 70B model information) fetched from the `/stats` endpoint, along with suggested questions for quick access. The main chat panel renders user questions and AI-generated answers with markdown formatting, source citation cards showing video title and timestamp range, relevance scores, and response time. The evaluation dashboard, accessible via a dedicated tab in the top navigation, presents system performance through three sub-tabs: a *Metrics* tab with four performance indicators displayed as progress bars, a *Comparison* tab with side-by-side RAG versus baseline bar charts, and a *System Info* tab listing complete configuration details. The frontend communicates with the backend exclusively through the Axios HTTP client library.

E. Evaluation Framework

A custom evaluation framework was implemented to assess system performance on 20 domain-specific DBMS questions [8], [26]. Each question is associated with a set of expected domain keywords. Four evaluation metrics are computed: Keyword Coverage measures the proportion of expected keywords present in the generated answer; Answer Completeness assigns a score based on answer word count thresholds; Source Relevance computes the average cosine similarity score of retrieved chunks; and Timestamp Coverage is a binary metric indicating

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

whether the generated answer contains at least one video timestamp citation. The overall score is the arithmetic mean of all four metrics. Baseline evaluation employs direct chunk retrieval without LLM generation, serving as the comparative reference for assessing RAG improvement.

VI. Experimental Evaluation

A. Evaluation Setup

Evaluation was conducted on a curated set of 20 domain-specific DBMS questions covering all major topic areas present in the lecture corpus. The questions span six categories: relational keys (primary key, candidate key, super key, foreign key), entity-relationship modeling (ER model, attribute types, relationship types), normalization (1NF, 2NF, 3NF, BCNF, functional dependency), SQL (DDL commands, DML commands, aggregate functions, joins, nested queries), transaction control (ACID properties, conflict serializability, two-phase locking), and indexing (B+ tree, single-level indexing). All experiments were conducted on CPU-only hardware running Python 3.11 on Windows, without any GPU acceleration, demonstrating the practical deployability of the system in resource-constrained environments.

B. Evaluation Metrics

Following established evaluation practices for RAG systems [8], [26], four metrics were used to evaluate system performance.

Keyword Coverage (KC): The proportion of domain-specific expected keywords present in the generated answer:

$$KC = \frac{\text{keywords found in answer}}{\text{total expected keywords}} \tag{1}$$

Answer Completeness (AC): A score based on answer word count reflecting response detail:

$$AC = \begin{cases} 0.3 & \text{if words} < 50 \\ 0.6 & \text{if } 50 \leq \text{words} < 100 \\ 0.8 & \text{if } 100 \leq \text{words} < 200 \\ 1.0 & \text{if words} \geq 200 \end{cases} \tag{2}$$

Source Relevance (SR): The average cosine similarity score of the top-k retrieved chunks with respect to the query [4]:

$$SR = 1 - \frac{1}{k}\sum_{i=1}^{k} d_i \tag{3}$$

where d_i is the cosine distance of the i-th retrieved chunk.

Timestamp Coverage (TC): A binary metric indicating whether at least one video timestamp is cited in the generated answer:

$$TC = \begin{cases} 1.0 & \text{if timestamp present in answer} \\ 0.0 & \text{otherwise} \end{cases} \tag{4}$$

The Overall Score (OS) is the arithmetic mean of all four metrics:

$$OS = \frac{KC + AC + SR + TC}{4} \tag{5}$$

C. Quantitative Results

Table II presents the comparative evaluation results of the proposed RAG system against the baseline keyword retrieval approach across all four metrics. The RAG system achieves an overall score of 88.0%, representing a +22.2% relative improvement over the baseline score of 72.1%. The most notable gains are Keyword Coverage (+32.0 pp) and Timestamp Coverage (+100.0 pp), consistent with the evaluation dashboard shown in the deployed application.

TABLE II
Evaluation Results: RAG System vs. Baseline

Metric	Base-line	RAG System	Gain (pp)
Keyword Coverage	47.2%	79.2%	+32.0
Answer Completeness	95.0%	99.0%	+4.0
Source Relevance	74.0%	74.0%	0.0
Timestamp Coverage	0.0%	100.0%	+100.0
Overall Score	**72.1%**	**88.0%**	**+15.9**
Avg. Response Time	1.10 s	6.62 s	—

pp = percentage points absolute gain; relative improvement = +22.2%.

D. Per-Category Analysis

Table III presents per-category performance scores for the RAG system across the six topic areas evaluated. Scores are reported on a 0–1 scale (1.0 = 100%).

TABLE III
Per-Category RAG System Performance (0–1 Scale)

Topic Category	Score	Observation
Relational Keys (PK, CK, FK)	0.955	Highest coverage
SQL Commands (DDL, DML)	0.910	Strong performance
Normalization (1NF–BCNF)	0.875	Good coverage
Transaction Control / ACID	0.842	Adequate
ER Model / Relationships	0.820	Moderate
Attribute Types (ER Model)	0.653	Weakest

E. System Configuration Summary

Table IV records the key system configuration parameters as reported in the deployed application's System Info tab, providing full reproducibility context.

F. Discussion

The proposed RAG system demonstrates substantial improvements over the baseline across three of four evaluation metrics. The most significant gain is observed in Keyword Coverage (+32.0 percentage points), confirming that LLM-generated answers [12] incorporate domain-appropriate terminology more comprehensively than direct chunk retrieval.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

TABLE IV
SYSTEM CONFIGURATION PARAMETERS

Parameter	Value
RAG Response Time	6.62 s
Baseline Response Time	1.1 s
Test Questions	20
Knowledge Chunks	131
Lecture Videos	20
LLM Model	LLaMA 3.3 70B
Embedding Model	MiniLM-L6-v2
Vector Database	ChromaDB
Speech Recognition	Whisper Base
Framework	FastAPI + React

This finding aligns with the broader observation in the RAG literature [1], [2] that LLM generation substantially enriches retrieval-only approaches through linguistic generalization.

Timestamp Coverage of 100% represents the most distinctive contribution of the proposed system. The baseline system, which returns raw chunk text without LLM generation, cannot produce timestamp citations by design, resulting in a score of 0%. The proposed system consistently cites precise video timestamps in every generated response, directly addressing the core educational use case of enabling students to locate specific content within source lectures. This capability is entirely absent in existing educational RAG systems [7], [9].

Source Relevance scores are identical for both systems (74.0%), as both employ the same ChromaDB semantic retrieval mechanism [13]. This confirms that the retrieval component performs consistently regardless of the downstream generation approach [4], validating the modular design of the pipeline.

The higher average response time of the RAG system (6.62 s versus 1.10 s for baseline) is attributable to the additional LLM inference step via the Groq API. This represents an acceptable trade-off given the substantial quality improvements across other metrics. Response times in the range of 6–7 seconds are well within acceptable thresholds for educational question-answering applications [19], where users expect thoughtful, detailed responses rather than instantaneous replies.

Per-category analysis reveals performance variation across topic areas. Questions on well-covered topics such as foreign keys (0.948), candidate keys (0.955), and primary keys (0.950) achieve the highest scores, reflecting strong lecture coverage in the knowledge base. Questions on more abstract topics such as attribute types in the ER model (0.653) show lower scores, suggesting that cross-lingual transcription may introduce noise for nuanced technical terminology [14], [22]. This finding motivates future work on domain-adapted embedding models [27].

VII. CONCLUSION AND FUTURE WORK

This paper presented the complete implementation and evaluation of a Cross-Lingual Retrieval-Augmented Generation system for intelligent DBMS education. The proposed system successfully addresses two critical gaps in existing educational AI literature [7], [9]: the inability to process regional-language video content, and the absence of timestamp-aware retrieval enabling precise video navigation for students.

The system processes 20 Hindi DBMS lecture videos through a ten-stage, two-phase pipeline comprising audio extraction, cross-lingual transcription via Faster-Whisper [6], [21], timestamp-aware chunking [28], dense vector embedding using all-MiniLM-L6-v2 [20], and ChromaDB indexing [13], resulting in a knowledge base of 131 semantic chunks. At inference time, the RAG pipeline [1] retrieves the top-5 most relevant chunks and generates coherent, source-grounded answers using LLaMA 3.3 70B [12] via the Groq API. The complete system is deployed as a full-stack web application with a FastAPI backend and React frontend featuring an integrated evaluation dashboard.

Experimental evaluation on 20 domain-specific DBMS questions demonstrates that the proposed system achieves an overall score of 88.0%, outperforming baseline keyword retrieval by 15.9 percentage points (22.2% relative improvement). Notably, the system achieves 100% timestamp coverage across all responses—a capability entirely absent in the baseline—enabling students to navigate directly to the relevant moment in source lecture videos. The system operates entirely on CPU hardware, requiring no GPU infrastructure, validating practical deployability. The RAG response time of 6.62 s and embedding model (MiniLM-L6-v2) confirm the system is suitable for deployment on standard academic hardware.

The key contributions are: a practical cross-lingual RAG pipeline for educational video content; a timestamp-aware chunking mechanism preserving temporal metadata throughout the retrieval pipeline; a custom four-metric evaluation framework for educational RAG systems [8], [26]; and an end-to-end deployable web application demonstrating viability on CPU-only hardware.

Future work will explore several directions. First, extension of the knowledge base to the complete playlist of 140 lectures to assess scalability and retrieval performance at larger corpus sizes, consistent with the scalability considerations identified in [2], [23]. Second, investigation of domain-adapted embedding models [27] trained on technical educational content to improve retrieval precision for abstract concepts. Third, integration of re-ranking mechanisms [3] to further improve retrieved chunk relevance. Fourth, extension to additional subjects and regional languages beyond Hindi [14], [22] to validate the generalizability of the cross-lingual pipeline. Fifth, incorporation of learner modeling [10] to personalize responses based on individual student knowledge levels and learning history. Finally, integration with knowledge graph representations [24] could enhance structured concept navigation within the DBMS domain.

REFERENCES

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-T. Yih, T. Rocktäschel, S. Riedel, and D. Kiela,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

"Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems (NeurIPS 2020)*, vol. 33, pp. 9459–9474, 2020.

[2] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," *arXiv:2312.10997*, 2023.

[3] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, and Q. Li, "A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models," in *Proc. 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: ACM, Aug. 2024, pp. 6491–6501. doi: 10.1145/3637528.3671470.

[4] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-T. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," in *Proc. 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pp. 6769–6781, 2020. doi: 10.18653/v1/2020.emnlp-main.550.

[5] C.-Y. Lin, K. Kamahori, Y. Liu, X. Shi, M. Kashyap, Y. Gu, R. Shao, Z. Ye, K. Zhu, R. Kadekodi, S. Wang, A. Krishnamurthy, L. Ceze, and B. Kasikci, "TeleRAG: Efficient Retrieval-Augmented Generation Inference with Lookahead Retrieval," *arXiv:2502.20969*, 2025.

[6] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," in *Proc. 40th International Conference on Machine Learning (ICML 2023)*, *PMLR*, vol. 202, pp. 28492–28518, 2023.

[7] U. H. Khan, M. H. Khan, and R. Ali, "Large Language Model Based Educational Virtual Assistant Using RAG Framework," *Procedia Computer Science*, vol. 252, pp. 905–911, 2025. doi: 10.1016/j.procs.2025.01.051.

[8] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "RAGAS: Automated Evaluation of Retrieval Augmented Generation," *arXiv:2309.15217*, 2023.

[9] R. Németh, A. Tátrai, M. Szabó, and Á. Tamási, "Using a RAG-Enhanced Large Language Model in a Virtual Teaching Assistant Role: Experiences from a Pilot Project in Statistics Education," *Hungarian Statistical Review*, vol. 7, no. 2, pp. 3–27, 2024. doi: 10.35618/hsr2024.02.en003.

[10] G. Yadav, Y.-J. Tseng, and X. Ni, "Contextualizing Problems to Student Interests at Scale in Intelligent Tutoring System Using Large Language Models," *arXiv:2306.00190*, May 2023.

[11] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al., "Llama 2: Open Foundation and Fine-Tuned Chat Models," *arXiv:2307.09288*, Jul. 2023.

[12] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, et al., "The Llama 3 Herd of Models," *arXiv:2407.21783*, Jul. 2024.

[13] J. J. Pan, J. Wang, and G. Li, "Survey of Vector Database Management Systems," *arXiv:2310.14021*, Oct. 2023.

[14] A. Piñeiro-Martín, C. García-Mateo, L. Docio-Fernández, M. del C. López-Pérez, and G. Rehm, "Weighted Cross-Entropy for Low-Resource Languages in Multilingual Speech Recognition," in *Proc. Interspeech 2024*, pp. 1235–1239. doi: 10.21437/Interspeech.2024-734.

[15] H. Zhang, J. Liu, Z. Zhu, S. Zeng, M. Sheng, T. Yang, G. Dai, and Y. Wang, "Efficient and Effective Retrieval of Dense-Sparse Hybrid Vectors Using Graph-Based Approximate Nearest Neighbor Search," *arXiv:2410.20381*, Oct. 2024.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," in *Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, Minneapolis, MN, Jun. 2019, pp. 4171–4186. doi: 10.18653/v1/N19-1423.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Proc. Advances in Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, 2017, pp. 5998–6008.

[18] C. Galli, N. Donos, and E. Calciolari, "Performance of 4 Pre-Trained Sentence Transformer Models in the Semantic Query of a Systematic Review Dataset on Peri-Implantitis," *Information*, vol. 15, no. 2, p. 68, Jan. 2024. doi: 10.3390/info15020068.

[19] Z. H. Sain, A. Vasudevan, and A. V. Lama, "The Emerging Future of AI Chatbots in Higher Education," *Jurnal Ilmiah Didaktika*, vol. 25, no. 1, pp. 93–107, Aug. 2024.

[20] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks," in *Proc. 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, Nov. 2019, pp. 3982–3992. doi: 10.18653/v1/D19-1410.

[21] D. Shah, R. Saboo, A. Dwivedi, and M. Gajjar, "Integrating Faster Whisper with Deep Learning Speaker Recognition," *International Journal of Computer Science and Mobile Computing*, vol. 13, no. 9, pp. 1–8, Sep. 2024. doi: 10.47760/ijcsmc.2024.v13i09.001.

[22] J. Wang, "Cross-Lingual Transfer Learning for Low-Resource Natural Language Processing Tasks," M.S. thesis, Inst. Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany, Feb. 2021.

[23] M. Cheng, Y. Luo, J. Ouyang, Q. Liu, H. Liu, L. Li, S. Yu, B. Zhang, J. Cao, J. Ma, D. Wang, and E. Chen, "A Survey on Knowledge-Oriented Retrieval-Augmented Generation," *arXiv:2503.10677*, 2025.

[24] L. Zhong, J. Wu, Q. Li, H. Peng, and X. Wu, "A Comprehensive Survey on Automatic Knowledge Graph Construction," *ACM Computing Surveys*, vol. 56, no. 4, Article 94, Nov. 2023. doi: 10.1145/3618295.

[25] M. F. ben Hassen, M. R. Bougherira, N. S. Alrayes, A. M. Alqahtani, N. Frih, and K. Kefi, "Enhancing Student Performance and Retention Through Synergistic Teacher and Peer-Recorded Videos in College Algebra," *SAGE Open*, Oct.–Dec. 2025. doi: 10.1177/21582440251400560.

[26] A. Farea, Z. Yang, K. Duong, N. Perera, and F. Emmert-Streib, "Evaluation of Question Answering Systems: Complexity of Judging a Natural Language," *ACM Computing Surveys*, vol. 58, no. 1, Article 1, Sep. 2025. doi: 10.1145/3744663.

[27] X. Zhang, K. Ogueji, X. Ma, and J. Lin, "Toward Best Practices for Training Multilingual Dense Retrieval Models," *ACM Transactions on Information Systems*, vol. 42, no. 2, Article 39, Sep. 2023. doi: 10.1145/3613447.

[28] C. A. Gomez-Cabello, S. Borna, S. M. Pressman, S. A. Haider, A. Genovese, B. G. Collaco, N. G. Wood, and A. J. Forte, "Comparative Evaluation of Advanced Chunking for RAG in Large Language Models for Clinical Decision Support," *Bioengineering*, vol. 12, no. 11, p. 1194, 2025. doi: 10.3390/bioengineering12111194.