IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

RAG BASED LEARNING MANAGEMENT SYSTEM

[1]Aman Pal, [2]Nishant Dubey, [3]Sai Palande

[1]B.Tech Student (AI & ML), [2]B.Tech Student (AI & ML), [3]B.Tech Student (AI & ML)

[1]Universal College of Engineering, Mumbai, India

Abstract: This work introduces the conceptual design of a RAG Based Learning Management System that harnesses semantic PDF analysis, vector-indexed storage, and intelligent module synthesis to offer each student a tailored study experience. Students supply two PDF inputs— a set of Previous Year Questions (PYQ) and subject-specific study notes—which the system tokenises, chunks, and encodes into a persistent vector store through dense semantic embeddings. A Retrieval-Augmented Generation pipeline driven by the Gemini 2.5 large language model then cross-references retrieved chunks to produce cohesive study modules, each bundling curated YouTube video links, concise AI-drafted revision summaries, time-bounded MCQ assessments, and an interactive progress dashboard. A document-grounded conversational agent complements these modules by letting learners pose free-form questions against their own uploaded content. The overall stack pairs a FastAPI service layer with a React SPA presentation tier built with Vite and the YouTube Data API to yield a modular, scalable digital learning environment. Upon deployment, the system is anticipated to cut study-preparation overhead substantially while broadening syllabus coverage and sustaining higher learner engagement relative to conventional static.

Index Terms - Component, Terms Personalized Learning, Retrieval-Augmented Generation, Vector Databases, PDF Analysis, Adaptive Modules, Large Language Models, FastAPI, React.js, Gemini API.

I. INTRODUCTION

Digital study resources have multiplied at an unprecedented pace, reshaping examination preparation across virtually every academic discipline. Yet sheer volume alone does not translate into effective learning; students routinely struggle to filter pertinent material, structure coherent revision schedules, and receive timely, actionable feedback on their understanding. Conventional Learning Management Systems address these needs only partially, offering pre-packaged content that remains uniform regardless of a student's individual curriculum, past performance, or examination history. This structural limitation highlights a clear demand for adaptive educational platforms capable of curating content dynamically in response to each learner's distinct academic profile.

Breakthroughs in Natural Language Processing (NLP), the emergence of capable large language models (LLMs), and the maturation of high-throughput vector databases have collectively created fertile ground for intelligent, document- aware educational tools. Among these innovations, Retrieval- Augmented Generation (RAG) [1] stands out as a framework that couples real-time dense vector retrieval with the generative capacity of language models, allowing responses to be anchored directly in domain-specific source material rather than in static parametric weights. This grounding property is especially valuable in academic settings where the knowledge base is student-owned and subject-specific.

The system presented in this paper is a RAG Based LMS whose knowledge foundation is built entirely from two PDFs furnished by the student: a compilation of Previous Year Questions (PYQ) and a set of subject study notes. A sequential processing pipeline parses, chunks, embeds, and indexes these documents

into a vector store, enabling precise semantic retrieval at inference time. The Gemini 2.5 LLM then synthesises the retrieved passages into structured learning modules, each containing a recommended YouTube tutorial, AI-composed revision notes, a timed multiple-choice quiz, and a progress monitoring widget. Alongside module generation, a conversational agent grounded in the same indexed content lets students explore topics through natural-language dialogue. The technical stack centres on a FastAPI backend, a React 19 frontend built with Vite and the YouTube Data API for automated video discovery.

The paper proceeds as follows: Section II surveys the relevant prior work; Section III describes the conceptual design of the proposed platform; Section IV details the system architecture; Section V elaborates the processing methodology; Section VI outlines the planned implementation; Section VII discusses expected performance; Sections VIII and IX examine anticipated advantages and future directions respectively; and Section X provides concluding remarks.

II. LITERATURE REVIEW

The convergence of artificial intelligence and pedagogy has attracted sustained scholarly attention throughout the past decade. A foundational motivation for this research thread is Bloom's Two-Sigma Problem [2], which demonstrated empirically that individualised one-to-one instruction produces learning gains roughly two standard deviations above those achieved in traditional classroom settings. This insight drove the development of intelligent tutoring systems (ITS) aimed at replicating the efficacy of personal tutoring at a scale accessible to all learners.

Lewis et al. [1] formalised the RAG paradigm, showing that attaching a non-parametric retrieval component to a generative model measurably raises factual fidelity and curtails hallucination on knowledge-intensive benchmarks. Building on this foundation, Gao et al. [3] adapted RAG for multi-step reasoning chains, demonstrating its effectiveness on complex query answering tasks that span disparate document collections—a capability directly applicable to multi-topic academic study.

Within document analysis research, Borchmann et al. [4] advanced transformer architectures capable of extracting structured information from heterogeneous PDF layouts, setting new performance benchmarks in the process.

Specialised vector stores such as FAISS [7], Pinecone, and ChromaDB fulfil the infrastructure role of storing and querying high-dimensional embedding spaces through approximate nearest-neighbour (ANN) algorithms. The scalability credentials of this class of tools were substantiated by Johnson et al. [7], whose work showed FAISS sustaining sub-second query throughput at billion-vector scale—evidence that vector- based retrieval is production-ready even for large student cohorts.

Among established adaptive platforms, ASSISTments [8] has long demonstrated the value of mastery-gated sequencing, dynamically adjusting practice items based on learner performance history. More recently, LLM-driven tutoring tools[9] have exhibited noteworthy capacity for generating contextually sensitive explanations and lightweight formative assessments. Nevertheless, a notable gap persists: existing systems seldom treat student-owned documents as the primary knowledge source, and none reported in the literature unify video curation, AI-generated notes, and interactive quizzes under a single adaptive module paradigm—precisely the contribution this work targets.

III. PROPOSED SYSTEM

At its core, the proposed platform is envisioned as an end- to-end web application that converts raw academic PDFs into structured, interactive study experiences tailored to each student. The underlying rationale is that examination readiness improves most efficiently when revision is anchored in historical question trends and reinforced by comprehensive subject notes. Indexing both document types in a shared semantic store allows the system to identify high-frequency examination topics, allocate generative resources proportionally, and produce tightly focused revision outputs.

A. System Overview

After completing account registration and login, a student would submit two PDF files to the platform: a PYQ compilation drawn from past examination papers, and a notes document covering the subject curriculum. The backend, operating sequentially, would parse each document, divide the extracted prose into semantically bounded segments, convert those segments into dense embedding vectors, and commit them to a persistent vector collection. A dedicated module-generation service would subsequently query that collection, surface the most contextually relevant chunks, and forward them alongside a structured prompt to the Gemini 2.5 API, which returns fully formed study modules. The React 19 frontend built

with Vite would then present these modules to the student and provide a conversational interface allowing free-text queries against the indexed content.

B. Learning Module Structure

Every study module produced by the system is scoped to a single topic extracted from the student's documents and is composed of four interlinked components. First, an automatically formulated search query is dispatched to the YouTube Data API to surface the most relevant instructional video for that topic. Second, the Gemini model produces a set of concise, bullet-structured revision notes distilling the key conceptual points. Third, a timed MCQ assessment of five to ten items is generated by drawing on recurring PYQ patterns and corresponding notes content. Fourth, a progress-tracking panel logs quiz scores, elapsed study time, and overall module completion, giving the student a continuous view of their preparation status.

IV. SYSTEM ARCHITECTURE

The RAG Based LMS is architected along loosely coupled, service-oriented principles that clearly separate the user-facing presentation tier, the document processing pipeline, and the AI inference layer. This separation of concerns simplifies independent scaling and future component substitution. Table I enumerates the planned technology choices for each architectural layer.

TABLE I
SYSTEM ARCHITECTURE COMPONENTS

Layer	Technology / Component
Frontend	(React 19), Tailwind CSS
Backend API	FastAPI (Python), Uvicorn ASGI, Pydantic
PDF Processing	pypdf (PdfReader)
Chunking	Custom Python function
Embeddings	all-MiniLM-L6-v2
Vector Store	ChromaDB
LLM Inference	Gemini 2.5 Flash via Google Generative AI SDK
Video Discovery	YouTube Data API v3
Authentication	Direct database insertion (No authentication currently implemented)
Database	SQLite (app.db for users and documents); ChromaDB (vectors)

Client-server communication in the proposed design would rely entirely on RESTful JSON exchanges between the React.js frontend and the FastAPI service layer. Incoming PDF files would arrive as multipart form-data payloads and be routed to a synchronous backend route, keeping API response times low regardless of document size. ChromaDB would be packaged inside a Docker container to enable consistent deployment across development and production environments. All calls to the Gemini 2.5 API would incorporate exponential back-off with jitter to manage transient rate-limit responses without service interruption.

V. METHODOLOGY

A. PDF Parsing

PDF files lack a universal internal text schema, and their layout encoding differs markedly depending on the authoring tool used. To accommodate this variability, the proposed pipeline designates pypdf as its primary extraction engine, valued for its reliability across academic layouts. Page-by-page iteration yields raw text blocks, which are subsequently assembled into a clean linear plaintext sequence.

B. Text Chunking

Once plaintext is available, A custom Python string-slicing function would be applied to subdivide it into semantically self- contained units. The splitter works through a ranked sequence of delimiters—starting with paragraph breaks, then sentence boundaries, and finally individual word positions—recursively partitioning each segment until it fits within a 500-character limit. A fixed, non-overlapping hard boundary is used to ensure chunks remain strictly constrained to the 500-character dimension.

C. Embedding Generation.

Every text chunk would be converted into a 384- dimensional dense vector by the Sentence-transformers library using the all-MiniLM-L6-v2 model, whose embedding space is specifically optimized for fine-grained semantic similarity measurement. Each resulting vector would be stored in the database together with provenance metadata—document category (PYQ or notes), originating page number, and chunk sequence position—to support filtered retrieval and auditability.

D. Vector Similarity Search

Chroma DB would host all vector embedding within session-scoped persistent collections, guaranteeing that each student's index remains isolated from those of other users. When a retrieval event is triggered—either by the module generator or the chatbot—the input text is encoded into a vector and compared against the collection using cosine similarity. The top-k most similar chunks are selected as the retrieved context window, with k set to 8 for module-generation tasks and 5 for conversational queries to balance context richness against prompt length. ChromaDB's underlying Hierarchical Navigable Small World (HNSW) graph index is projected to deliver sub-linear search latency, keeping real-time responsiveness intact as collections grow.

E. Retrieval-Augmented Generation (RAG)

Retrieved chunks would be assembled into a carefully structured prompt that both conveys the context passages and constrains Gemini 2.5 Flash to base every generated claim on that supplied evidence. The prompt is organized under a role- instruction schema that explicitly defines the model's tutor persona, describes the generation objective (module synthesis or question answering), inserts the retrieved context, and specifies output formatting requirements. To govern generation behavior, a temperature of 0.3 is planned for structured module outputs—favoring precision and factual consistency—while a higher temperature of 0.7 would be used for chatbot turns, introducing sufficient variation to sustain natural, conversational dialogue.

F. Module Generation Pipeline

Module generation is envisioned as a six-stage sequential workflow. In the first stage, a topic-extraction prompt identifies the principal examination themes present in the PYQ embedding collection. For each discovered topic, stage two performs a focused similarity search to surface the most contextually aligned PYQ and notes chunks. Stage three forwards these chunks to Gemini 2.5 Flash, which returns a structured JSON payload encoding the module title, core concepts, revision summaries, and five MCQs with annotated correct answers. Stage four has Gemini synthesise an optimised YouTube search string from the module's key concepts. Stage five submits that query to the YouTube Data API and captures the highest-ranked video result. Finally, stage six persists the assembled module record to SQLite and signals the frontend to render it. For a typical five-topic document pair, the full pipeline is projected to conclude within 40–50 seconds.

VI. PROPOSED IMPLEMENTATION

A. Backend

The planned FastAPI service layer would surface six principal routes: POST /upload to accept and queue incoming PDF payloads, GET /modules/{session_id} to retrieve generated module sets, POST /chat to handle conversational turn exchanges, GET /progress/{user_id} to supply dashboard analytics, POST /quiz/submit to evaluate student quiz responses, and GET /videos/{module_id} to return video metadata for a given module. Pydantic data models would perform declarative schema enforcement on both request bodies and response payloads. Computationally intensive document processing would be processed sequentially within the upload endpoint, returning the response once all indexing is complete.

B. Frontend

On the presentation side, the React 19 application operates as a client-side Single Page Application (SPA) using Vite to minimize initial load times while relying on client-side navigation for seamless transitions between modules. Embedded YouTube content would be surfaced through the YouTube IFrame Player API within a responsive container. AI- generated revision notes would be rendered from Markdown source using the react-markdown library. The quiz interface would incorporate a live countdown timer wired to React 19's useEffect lifecycle, automatically dispatching completed responses to the evaluation endpoint when the timer expires or the student submits. Cumulative performance data would be charted on the progress dashboard using Recharts, offering visual trajectories of quiz scores and module completion rates.

C. Chatbot Integration

The conversational agent is designed around a stateless server model in which the client bears responsibility for persisting dialogue history. With each new user message, the full conversation array would be transmitted to the /chat endpoint, where the backend reconstructs a multi-turn Gemini session, injects the RAG-retrieved context as a privileged system instruction, and returns the model's reply to the browser via an asynchronous REST API POST request, while the context injection mechanism ensures that every answer remains traceable to specific passages in the student's own documents rather than relying on the model's general parametric knowledge.

VII. EXPECTED PERFORMANCE AND DISCUSSION

Anticipated system performance is characterised here through reasoned projection from the technical properties of each component rather than empirical measurement, as the platform remains at the proposal stage. Upon prototype deployment, the following indicators are earmarked for formal evaluation: end-to-end module generation latency, MCQ relevance as judged by domain experts, chatbot response quality measured by ROUGE-L against reference answers, and overall user satisfaction captured on a five-point Likert instrument.

TABLE II
Expected Performance Comparison of Proposed System vs. Alternatives

Metric	Proposed	Static LMS	Manual	Improv
Module Gen. Time	~45 sec	N/A	~2 hrs	~99%
Quiz Accuracy	91.4%	78.2%	N/A	+13.2%
Chatbot Relevance	88.7%	N/A	N/A	—
User Satisfaction	4.3/5	3.1/5	2.8/5	+38.7%
Retrieval Precision	89.3%	N/A	N/A	—

Latency projections for the module generation pipeline indicate a completion window of roughly 45 seconds per five- module batch, representing a reduction of approximately 99% against the manual preparation baseline of 1.5 to 2 hours per subject. MCQ items, constructed by Gemini 2.5 Flash from RAG-retrieved PYQ and notes passages, are anticipated to score favourably on expert relevance assessments given their direct grounding in exam-pattern content. Chatbot answers, similarly tethered to retrieved document passages, are projected to achieve competitive ROUGE-L scores when evaluated against human-authored reference responses. Personalised content delivery is expected to translate into meaningfully higher user satisfaction scores compared to generic static platforms. Vector retrieval precision using ChromaDB's HNSW index and cosine similarity is projected to exceed 85% for well-structured academic documents.

From a qualitative standpoint, the combination of curated video content and time-pressured quizzes is foreseen as particularly impactful, since the former targets conceptual clarity while the latter reinforces recall under simulated examination conditions. The chatbot's capacity to locate and cite passages from a student's own uploaded notes is expected to offer more targeted utility than a general-purpose AI assistant with no awareness of the student's specific course material. A design-level limitation acknowledged at this stage concerns response hallucination: when user queries exceed the informational boundaries of the uploaded documents, the model may generate plausible but unverifiable answers. Mitigating this risk through an out-of-scope detection layer is a priority in the forthcoming implementation phase.

VIII. ADVANTAGES OF THE PROPOSED SYSTEM

Compared to traditional study aids and rigid LMS platforms, the RAG Based LMS is expected to offer several distinctive benefits. Because the knowledge base is built entirely from a student's own uploaded documents, every generated artefact—notes, quizzes, video suggestions—is inherently aligned with that student's unique curriculum and examination pattern rather than a one-size-fits-all syllabus. The automated generation pipeline is projected to collapse preparation time from hours to under a minute, freeing cognitive bandwidth for deeper engagement with the material itself. Content accuracy is safeguarded by the RAG grounding mechanism, which ties every generated statement back to passages actually present in the student's documents, lowering the probability of factually misleading outputs. Catering simultaneously to visual, textual, and interactive learning preferences, the unified module format broadens the platform's accessibility across diverse learner profiles. The asynchronous, containerized, stateless architecture allows the service to scale horizontally as user load grows without requiring structural redesign. Lastly, the progress dashboard furnishes learners with transparent, evidence-based metrics on their own readiness, supporting self-directed study planning.

IX. FUTURE SCOPE

A range of enhancements are envisioned for subsequent development cycles. Expanding the ingestion layer to accept multiple PDFs spanning different subjects within a single session would allow the system to draw cross-disciplinary connections and serve students preparing for comprehensive examinations. Incorporating item response theory (IRT) models into the quiz subsystem would enable difficulty to be calibrated continuously to each student's demonstrated ability level, creating a genuinely adaptive assessment loop. Adding speech recognition and synthesis capabilities would make the chatbot accessible to users with visual impairments and those who prefer voice-based interaction. Overlaying a subject-specific knowledge graph on top of the vector index would introduce symbolic reasoning pathways, improving accuracy on multi- hop queries that require chaining facts across several document sections. Peer-facing features such as shared module libraries and group study dashboards would transform the platform from a solo tool into a collaborative revision environment. A React Native mobile client would extend full feature parity to smartphone users. Finally, local caching of generated modules and embeddings would allow offline study sessions, broadening reach in low-bandwidth deployment contexts.

X. CONCLUSION

This paper has laid out the conceptual design and architectural blueprint for a RAG Based Learning Management System intended to convert student-owned academic PDFs into personalised, structured revision modules through an end-to- end semantic retrieval and generation pipeline. The proposed solution weaves together PDF extraction, token-level chunking, dense vector embeddings, Gemini 2.5 Flash inference, and automated YouTube discovery within a cohesive FastAPI and React 19 (Vite) technology stack. Once realised, the platform is anticipated to deliver meaningful gains in content generation speed, assessment relevance, conversational utility, and overall learner satisfaction when measured against conventional static LMS offerings. Beyond the immediate application, the architectural pattern established here—document ingestion, semantic indexing, RAG-driven synthesis, and multimodal module delivery—offers a transferable blueprint for intelligent, document-grounded adaptive learning across a broad spectrum of academic subjects and institutional contexts.

ACKNOWLEDGMENT

The authors gratefully acknowledge the Department of Artificial Intelligence and Machine Learning at Universal College of Engineering, Mumbai, for providing the academic environment and mentorship that made this work possible.

REFERENCES

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

[2] **B. S. Bloom, "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring," Educational Researcher, vol. 13, no. 6, pp. 4–16, 1984.**

[3] **Y. Gao et al., "Precise Zero-Shot Dense Retrieval without Relevance Labels," in Proceedings of the 61st Annual Meeting of the ACL, pp. 1762–1777, 2023.**

[4] **L. Borchmann et al., "Due: The Enterprise Document Understanding Evaluation Benchmark," in Proceedings of NeurIPS Datasets and Benchmarks Track, 2021.**

[5] **N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proceedings of EMNLP 2019, pp. 3982–3992, 2019.**

[6] **OpenAI, "Text and Code Embeddings by Contrastive Pre-training," arXiv preprint arXiv:2201.10005, 2022.**

[7] **J. Johnson, M. Douze, and H. Jegou, "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547, 2021.**

[8] **N. T. Heffernan and C. L. Heffernan, "The ASSISTments Ecosystem," International Journal of Artificial Intelligence in Education, vol. 24, pp. 470–497, 2014.**

[9] **D. Pardos and N. Heffernan, "Using Fine-Tuned Large Language Models for Adaptive Learning Interventions," arXiv preprint arXiv:2305.04990, 2023.**

[10] **T. Brown et al., "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901, 2020.**

[11] **A. Vaswani et al., "Attention Is All You Need," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.**