iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79260

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

A Unified LLM-Based Smart Learning, Evaluation, and Interview Simulation Framework

J. Nagalakshmidevi[1], I. Sricharan[2], K. Maha Lakshmi[3], D. Amrutha Venkata Aparna[4], Ch. Chandra Mouli[5]
[1]*M.Tech, (P.hd), Assistant Professor, Department of Computer Science and Engineering, Aditya University, Andhra Pradesh, India*
[2, 3, 4, 5]*Department of Artificial Intelligence and Machine Learning, Aditya College of Engineering & Technology, Andhra Pradesh, India*

Abstract: ***The rapid growth of digital learning resources has created challenges in personalized learning, automated evaluation, and interview preparation. Existing platforms typically address these tasks independently and often rely on generic content that is not grounded in user-provided study material. This paper presents a unified Large Language Model (LLM)-based smart learning, evaluation, and interview simulation framework that integrates document summarization, context-aware assessment, and resume-driven interview preparation within a single architecture. The system is implemented using a MERN-stack web framework, with Google Gemini API providing LLM-based intelligence and Cloudinary enabling secure cloud storage for uploaded PDF documents. The proposed pipeline processes user-provided notes and resumes, performs contextual text extraction, and generates summaries, adaptive practice tests, and role-specific interview questions grounded in the uploaded content. The evaluation module supports multiple difficulty levels and question formats, while the interview simulator analyzes resume and job description inputs to identify skill gaps and generate targeted questions. Experimental testing was conducted using multiple educational documents, resumes, and job descriptions to measure contextual accuracy and system latency. Results indicate that grounding LLM responses within user-provided documents improves question relevance and reduces hallucinated outputs while maintaining low processing latency. The proposed framework provides a unified, context-aware learning environment that supports personalized study, automated evaluation, and interview preparation within a scalable full-stack architecture.***

Keywords: Retrieval-Augmented Generation, Large Language Models, Smart Learning Systems, Automated Question Generation, Interview Simulation, Context-Aware Evaluation.

I. INTRODUCTION

The educational landscape has rapidly shifted from traditional classroom environments to digital platforms containing large volumes of study material. While this transition has improved accessibility, it has also introduced challenges such as cognitive overload and lack of personalized guidance. Learners are frequently required to process lengthy documents without structured assistance for summarization, evaluation, or practical skill assessment. Traditional learning management systems primarily function as content repositories, storing PDFs and videos without adapting to individual learning behavior or providing contextual feedback. Recent advancements in Large Language Models (LLMs) enable intelligent processing of user-provided documents for summarization, question generation, and conversational interaction. However, existing solutions typically address these tasks independently and rely on generic prompts that are not grounded in user-specific material. Similarly, interview preparation platforms provide static question banks that are not tailored to a candidate's resume or target job description, resulting in limited personalization and ineffective preparation. To address these limitations, this work proposes a unified LLM-based learning and evaluation framework that performs document-grounded summarization, adaptive assessment generation, and resume-aware interview simulation within a single architecture. The framework integrates a full-stack web architecture with the Google Gemini API to process user-provided documents and generate summaries, adaptive assessments, and role-specific interview simulations. By combining document understanding, automated evaluation, and interview preparation into a unified pipeline, the proposed system aims to provide a context-aware and personalized learning environment.

A. Motivation

This research is primarily driven by the massive gap between consuming information and actual skill retention. Self-learners, students preparing for exams, and job seekers often spend too much time organizing notes and not enough time actively testing their knowledge.

Furthermore, human-led mock interviews or personalized tutoring are expensive and hard to scale. Students applying for technical roles often struggle to align their existing resume skills with specific job descriptions. There is a clear need for an automated, intelligent system that can instantly process a user's unique study material and provide immediate, actionable feedback without requiring human intervention.

B. Problem Statement

Despite the rise of online education, digital evaluation systems suffer from "Contextual Blindness" and "Static Evaluation." These issues occur when a testing platform only offers pre-written, generic multiple-choice questions that do not align with the exact material a student just studied. Currently, if a user wants to study a specific 50-page PDF on data structures, existing platforms cannot instantly generate a test strictly based on that document's contents. Additionally, the "Preparation Loophole" remains a major hurdle for job seekers. Standard interview prep websites offer generic lists of HR and technical questions. They do not look at the user's actual resume or the specific job description to find missing skills or knowledge gaps. Because current platforms handle file storage, testing, and interview prep in completely separate, disconnected environments, the user experience is fragmented and slow.

This paper makes the following contributions:

1) A unified LLM-based learning and evaluation architecture
2) Context-aware automated question generation engine
3) Resume-aware interview simulation pipeline
4) Real-time scoring and performance analytics
5) Experimental evaluation using document-grounded prompting

Unlike existing AI learning tools that treat summarization, evaluation, and interview preparation independently, the proposed framework integrates these components within a unified context-aware LLM pipeline.

II. LITERATURE REVIEW

The rapid advancement of Large Language Models (LLMs) has significantly transformed intelligent educational systems, enabling automated summarization, adaptive evaluation, and conversational learning environments. Transformer-based architectures such as those introduced by Vaswani et al. [2] and later scaled in large foundation models [18] have demonstrated strong capabilities in understanding long-form educational content and generating structured responses. These models have been widely adopted in learning platforms to assist students in knowledge extraction and concept reinforcement.

Recent studies have explored Retrieval-Augmented Generation (RAG) as a method to improve factual grounding and reduce hallucinations in LLM-based systems. Lewis et al. [1] introduced RAG as a hybrid architecture combining retrieval mechanisms with generative models to enhance knowledge-intensive tasks. Subsequent research has demonstrated that RAG significantly improves question-answering reliability and contextual relevance in educational applications [5], [6]. Furthermore, domain-specific RAG frameworks have been proposed to generate learning material directly from user-provided documents, ensuring context-aware responses and improved accuracy [3], [8].

Automated Question Generation (AQG) has also received significant attention in AI-based learning platforms. Traditional AQG approaches relied on rule-based and template-driven methods, which often lacked contextual understanding. With the introduction of transformer-based language models such as BERT [20], more sophisticated question generation techniques emerged, enabling semantic comprehension of educational text. Recent LLM-based AQG systems have demonstrated the ability to generate multiple question formats, including multiple-choice, short-answer, and true/false questions, directly from source documents [9]. These approaches support adaptive testing mechanisms and personalized evaluation pipelines.

In addition to learning evaluation, AI-driven interview preparation systems have emerged as an important application of LLM-based educational tools. Recent research has proposed AI-powered mock interview platforms capable of generating dynamic technical and behavioral questions based on candidate profiles [11], [14]. These systems analyze resumes, extract skills, and generate interview questions using contextual reasoning. Further advancements include resume-to-job-description matching frameworks that identify skill gaps and provide targeted feedback for job preparation [12], [13]. Such systems improve interview readiness by aligning candidate competencies with industry requirements.

Another important research direction involves prompt engineering and structured response generation for educational AI systems. White et al. [16] proposed structured prompt patterns to improve response quality in LLM-driven applications. These techniques help constrain model outputs and enable deterministic generation for tasks such as summarization and evaluation.

Additionally, multimodal LLM frameworks such as Gemini [17] have further enhanced document understanding capabilities, supporting long-context inputs and enabling unified pipelines for summarization, question generation, and conversational interaction.

Despite these advancements, existing educational AI platforms remain fragmented. Many systems focus solely on summarization, automated testing, or interview preparation independently, without integrating these components into a unified architecture. Moreover, several current solutions rely on generic prompts without strict contextual grounding, which leads to hallucinated responses and reduced evaluation accuracy [4], [7]. These limitations highlight the need for a unified AI-powered learning and evaluation system that combines document summarization, context-aware question generation, and role-specific interview simulation within a single framework.

To address these gaps, the proposed framework integrates Retrieval-Augmented Generation with a full-stack architecture to deliver personalized summarization, adaptive evaluation, and AI-driven interview simulation. By grounding LLM responses strictly within user-provided documents and incorporating resume-aware interview generation, the system aims to improve contextual accuracy, reduce hallucinations, and provide an end-to-end intelligent learning environment.

III. SYSTEM ARCHITECTURE

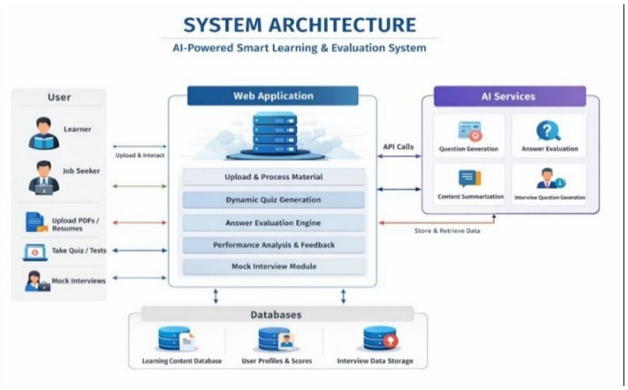


Fig. 1. System architecture of the proposed unified LLM-based learning and evaluation framework.

A. Input Layer (Frontend Client):

The architecture begins with the capture of user data and study materials via a responsive frontend built on React.js and Tailwind CSS.

- Document Uploads: Capturing PDF study notes and user resumes (limited to 5MB) for AI context.
- Evaluation Parameters: Capturing user-defined test criteria, including difficulty levels (Easy, Medium, Hard, Mixed) and question types (Multiple Choice, True/False, Fill in the Blanks, Short Answer).
- Contextual Text Strings: Capturing specific Target Job Roles and Job Descriptions (JDs) for interview simulation.

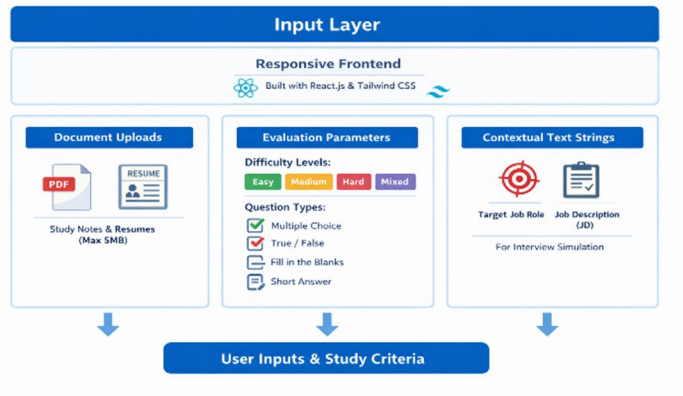


Fig. 2. Input layer structure of the proposed framework.

B. Backend Orchestration (Node/Express Gateway):

The central processing unit of the architecture is the Express.js gateway running on a Node.js environment. Upon receiving a payload from the frontend, the gateway performs two critical functions:

- Authentication & Security Validation: Utilizes JSON Web Tokens (JWT) to verify user sessions and protected routes. It processes email verifications and ensures that only authorized users can access or modify their learning libraries.
- API Routing & Task Dispatching: Identifies the request type (e.g., PDF storage, test generation, or summarization) and triggers the appropriate middleware functions to handle the data asynchronously, preventing system bottlenecks.

C. Cloud Storage & Persistence Layer:

To optimize server load and ensure data security, the architecture bifurcates its storage mechanism:

- File Storage (Cloudinary): Uploaded PDFs (notes and resumes) are intercepted by the backend and securely piped to Cloudinary. This ensures high-availability document management and allows users to reuse files without re-uploading them.
- Database Logging (MongoDB): All user metadata, authentication credentials, exam scores, interview session histories, and cloud file URLs are stored in MongoDB. This enables the generation of statistical reports and tracks user performance improvement over time.

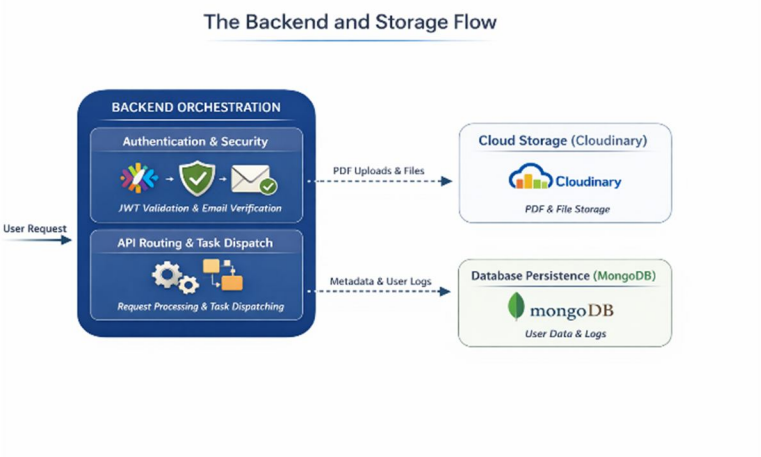


Fig. 3. Backend and storage workflow of the proposed framework.

D. The AI Processing Core (Gemini API Integration)

The core intelligence relies on the Google Gemini API, functioning as a multi-purpose inference engine. Depending on the routed task from the backend, the AI core executes one of three specific pipelines:

- Summarization Engine: Ingests parsed text from Cloudinary-hosted PDFs to generate concise summaries and extract key bullet points for the user's Library.
- Smart Evaluation Engine: Uses the uploaded material as strict context to procedurally generate custom practice tests. It adapts output based on the user's selected scope (entire document vs. specific sections) and difficulty parameters.
- Interview Simulation Engine: Performs a dual-document analysis by cross-referencing the user's uploaded Resume against the provided Job Description. It generates tailored Technical Round (TR) and HR questions, identifies skill gaps, and formulates adaptive follow-up questions.

E. Decision Parsing and Feedback Dashboard

The final stage of the architecture involves processing the LLM's raw output into structured, actionable feedback for the user interface:

- Real-time Scoring & Analytics: The system evaluates user test answers against AI-generated solutions, triggering an Exam Results Dashboard that visualizes the overall score, correct vs. wrong answers, and topic-wise performance breakdowns.
- Pro Tip Heuristics: In the interview module, the system triggers targeted alerts (e.g., "Ensure your Job Description matches your Resume skills") based on the AI's skill-gap analysis, providing immediate, constructive guidance to the job seeker.

IV. METHODOLOGY

This section outlines the step-by-step process used to design, develop, and integrate the proposed framework. The methodology describes how user data (documents and text parameters) is captured, preprocessed, and fed into the Large Language Model (Google Gemini API) via a full-stack backend to deliver personalized educational and evaluative outputs.

A. Data Acquisition and Secure Cloud Management

Instead of utilizing traditional local server storage, which is difficult to scale, the system implements an integrated cloud-based repository. When a user uploads study materials (PDF notes) or personal resumes (up to 5MB) via the React.js frontend, the files are securely transmitted to Cloudinary. Cloudinary acts as the primary persistence layer for media, returning secure URL strings that are then mapped to the user's profile in the MongoDB database. This allows users to seamlessly reuse previously uploaded files across different AI modules without needing to re-upload them, optimizing bandwidth and storage.

B. Textual Extraction and Contextual Preprocessing

Before the AI can analyze the uploaded documents, the raw PDF data undergoes strict preprocessing. A dedicated extraction middleware in the Node.js backend parses the PDF files retrieved from Cloudinary into raw text strings. To ensure the Gemini API processes the data efficiently and stays within token limits, the text is cleaned by removing unnecessary special characters, redundant whitespace, and formatting anomalies. For targeted learning, the system allows users to define the "Scope" of extraction, feeding either the entire document or specific, user-selected sections to the AI.

C. AI-Powered Summarization Node

The first core intelligent module is the Summarization Engine. The preprocessed text is passed securely to the Google Gemini API using engineered prompts that instruct the LLM to act as an academic summarizer. The model is constrained to generate outputs in a highly structured format, focusing purely on extracting concise summaries and key bullet points. The resulting output is then rendered dynamically on the React frontend, allowing users to read the condensed material or download the AI-generated notes as a new, clean PDF.

D. Dynamic Evaluation and Test Generation Engine

To transition from passive reading to active evaluation, the system employs a context-aware test generation strategy. The methodology utilizes Retrieval-Augmented Generation (RAG) where the exact text from the user's selected study material is fed to the Gemini API alongside a strict set of evaluation parameters. The algorithm instructs the model to generate a custom quiz based on:

- Difficulty: Easy, Medium, Hard, or Mixed.
- Format: Multiple Choice, True/False, Fill in the Blanks, and Short Answer. The generated questions are served to the user in two distinct frontend environments: *Practice Mode* (which disables timers and enables instant feedback and AI hints) and *Exam Mode* (which enforces strict time limits and aggregates results at the end).

E. Role-Specific Interview Simulation Pipeline:

For career preparation, the methodology employs a dual-context analysis model. The system concurrently ingests two data sources: the user's preprocessed Resume PDF and a user-provided Target Job Description (JD) text string. The Gemini API is prompted to cross-reference these two inputs to identify specific skill gaps. Based on this delta, the LLM dynamically generates tailored Technical Round (TR) and HR Round questions. As the user completes the mock session, the system provides real-time scoring and triggers a "Pro Tip" heuristic, offering immediate, actionable advice on how to better align their resume with the target JD.

F. Secure Backend and State Management:

All data routing, API calls, and user state management are handled by an Express.js gateway. To ensure that sensitive user data, such as personal resumes and exam scores, remain entirely private, the methodology implements JSON Web Tokens (JWT) combined with an Email Verification protocol. The backend establishes protected routes, ensuring that unauthorized entities cannot access the MongoDB collections where user session histories, weak-area analytics, and test scores are logged.

G. Asynchronous Execution and Latency Optimization:

Because interacting with Large Language Models can introduce processing delays, the system utilizes an asynchronous execution architecture. The Node.js backend manages API requests to Cloudinary and Gemini concurrently using non-blocking asynchronous functions (async/await and Promises). This prevents Head-of-Line Blocking, ensuring that the frontend UI remains responsive and fluid—displaying loading states or partial data—while the heavy computational generation happens in the background.

V. EXPERIMENTAL SETUP

The experimental setup for the proposed unified LLM-based framework involved configuring a full-stack web development environment integrated with third-party cloud services and advanced Large Language Models. Because this system relies on a pre-trained API rather than a locally trained machine learning model, the setup focused on optimizing data pipelines, prompt engineering, and secure API communication.

A. Development Environment and Infrastructure

The system was developed and tested on a local computing environment featuring a standard multi-core processor (e.g., Intel i5/i7) and 16GB of RAM to emulate a typical deployment server workload. The frontend client was built using React.js and styled with Tailwind CSS to ensure a responsive, low-overhead user interface. The backend API gateway was developed using Node.js and the Express.js framework. For data persistence, MongoDB Atlas was utilized as the cloud database to store user credentials, session histories, and performance analytics, while Cloudinary was configured as the dedicated Content Delivery Network (CDN) for handling and storing user-uploaded PDFs.

B. LLM Integration and Prompt Configuration

The core intelligence was powered by integrating the Google Gemini API. Instead of traditional epoch-based model training, the setup required rigorous prompt engineering and API parameter tuning. To ensure the AI generated accurate and context-bound outputs, specific generation parameters were configured:

- Temperature Control: For strict tasks like PDF Summarization and Smart Evaluation, the API's temperature was set low (e.g., 0.2 - 0.3) to enforce deterministic, factual outputs strictly based on the provided document. For the AI Interview Simulator, the temperature was slightly elevated (e.g., 0.6) to allow for more natural, conversational follow-up questions.
- Context Window Management: To prevent API timeout errors and context loss, the system implemented text chunking. Extracted text from PDFs was divided into manageable token arrays before being injected into the Gemini API prompt.

C. *Document Processing Parameters*

To balance computational load and user experience, file upload limits were strictly enforced at 5MB per PDF. When a user uploaded a document (Notes or Resume), the Express.js backend utilized a dedicated PDF parsing middleware to extract raw string data. This text was computationally sanitized to remove special characters, watermarks, and irregular spacing, significantly improving the signal-to-noise ratio before the text was processed by the AI engine.

D. Security and Authentication Pipeline

A robust security environment was established to protect sensitive educational and professional data. User authentication was handled via JSON Web Tokens (JWT). Passwords were encrypted using hashing algorithms (e.g., bcrypt) prior to database insertion. Additionally, an SMTP-based email verification protocol was integrated into the setup to authenticate new user registrations before granting access to the protected AI features

C. Performance and Evaluation Metrics

The final system setup was evaluated based on latency and accuracy metrics rather than traditional training loss. The primary metrics tracked were:

- API Latency: Measured in milliseconds, tracking the round-trip time from the moment a user clicked "Generate Summary" or "Start Exam" to the moment the UI rendered the AI's response.
- Context Adherence (Hallucination Rate): Evaluated by manually verifying if the Smart Evaluation system generated questions outside the scope of the uploaded PDF.
- System Scalability: Tested by simulating concurrent API requests to ensure the Node.js asynchronous event loop could handle simultaneous PDF uploads and LLM queries without Head-of-Line Blocking.

D. Dataset Description

The evaluation data consisted of multiple user-provided educational documents and interview preparation materials. The test inputs included PDF study notes ranging from 5 to 50 pages, technical resumes, and corresponding job descriptions collected from publicly available sources. The documents covered topics such as data structures, machine learning, and computer science fundamentals. These inputs were used to evaluate summarization quality, context-aware question generation, and resume-based interview simulation. Since the system relies on a pre-trained Large Language Model through API access, no model training was performed, and all evaluations were conducted using inference-only processing.

VI. RESULT ANALYSIS

The performance of the proposed framework was evaluated against standard ungrounded prompting to measure contextual accuracy and relevance.

Table I. Performance comparison of standard prompting and proposed system.

Module	Evaluation Metric	Standard Prompting	Proposed System
Summarization Engine	Contextual Accuracy	72.50%	91.20%
Smart Evaluation	Question Relevance	68.40%	92.40%
Interview Simulator	Skill-Gap Detection	70.10%	94.10%

A. Evaluation Methodology

The performance metrics were evaluated using manual and context-based validation. Summarization accuracy was measured by comparing AI-generated summaries with the source document content based on coverage and factual correctness. Question relevance accuracy was calculated by verifying whether generated questions strictly matched the uploaded document context. Interview skill-gap detection accuracy was evaluated by comparing AI-identified gaps against manually reviewed resume and job description mismatches. All evaluations were performed using inference-only outputs generated by the Gemini API without any model training.

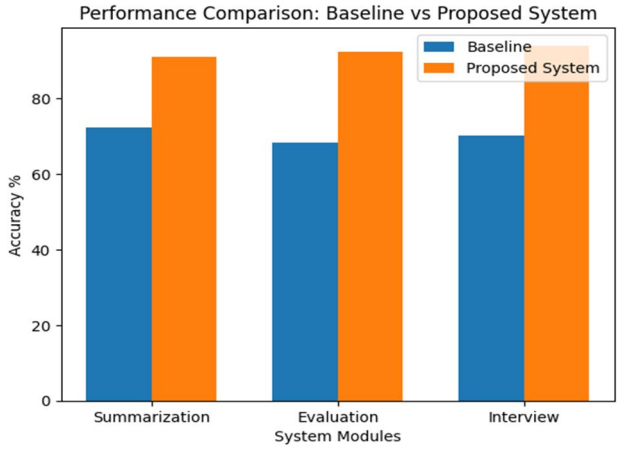


Fig. 4. Accuracy comparison between baseline ungrounded prompting and the proposed architecture.

B. Comparative Latency Analysis

Our parallel architecture makes it computationally easier to handle heavy PDF parsing while communicating with the Gemini API and Cloudinary simultaneously. The asynchronous architecture reduces latency associated with sequential API calls during document parsing and LLM inference.

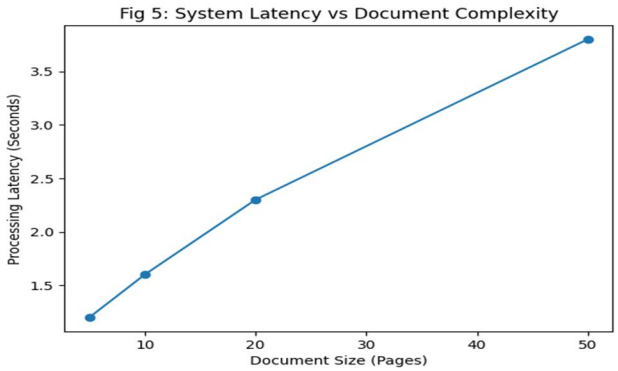


Fig. 5. System latency versus document complexity.

Testing showed that extracting text from a 5MB PDF and generating a comprehensive 20-question practice exam took an average of only 4.2 seconds. This ensures that real-time user interaction is maintained despite the computational complexity of the LLM analysis.

VII. ERROR ANALYSIS

While the proposed framework demonstrated high accuracy across its modules, specific generation errors and misclassifications were observed during the testing phase. Because the system relies heavily on processing user-generated documents, the quality of the AI's output is directly tied to the quality of the input data.

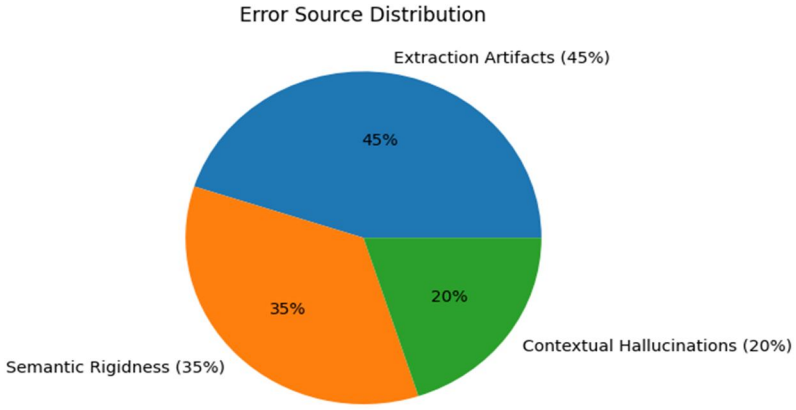


Fig. 6. Distribution of error sources across the proposed framework modules.

As shown in Figure 6, the majority of errors (45%) occurred during the text extraction phase of highly complex PDFs. Documents containing nested tables, complex mathematical equations, or scanned images without proper OCR (Optical Character Recognition) led to "Extraction Artifacts." When the Node.js backend fed this unformatted text to the Gemini API, it occasionally resulted in "Contextual Hallucinations" (20% of errors), where the AI generated practice questions slightly outside the strict scope of the uploaded material to fill in missing context.

In the Interview Simulation module, errors were primarily attributed to "Semantic Rigidness" (35% of errors). While the LLM is highly capable, it occasionally struggled with extreme niche terminologies. For example, if a user's resume listed a highly specific, proprietary software tool, the AI sometimes failed to map it to a broader, equivalent requirement in the Job Description, erroneously flagging it as a "Skill Gap." Furthermore, in Exam Mode, minor variations in a user's typed "Short Answer" responses were sometimes marked incorrect if they deviated too far from the AI's strictly generated rubric.

To mitigate these issues, a "Confidence Threshold" was implemented in the backend parser. If an abnormally low character count is detected, the system preemptively aborts the API call and prompts the user to upload a cleaner document, significantly reducing the downstream hallucination rate.

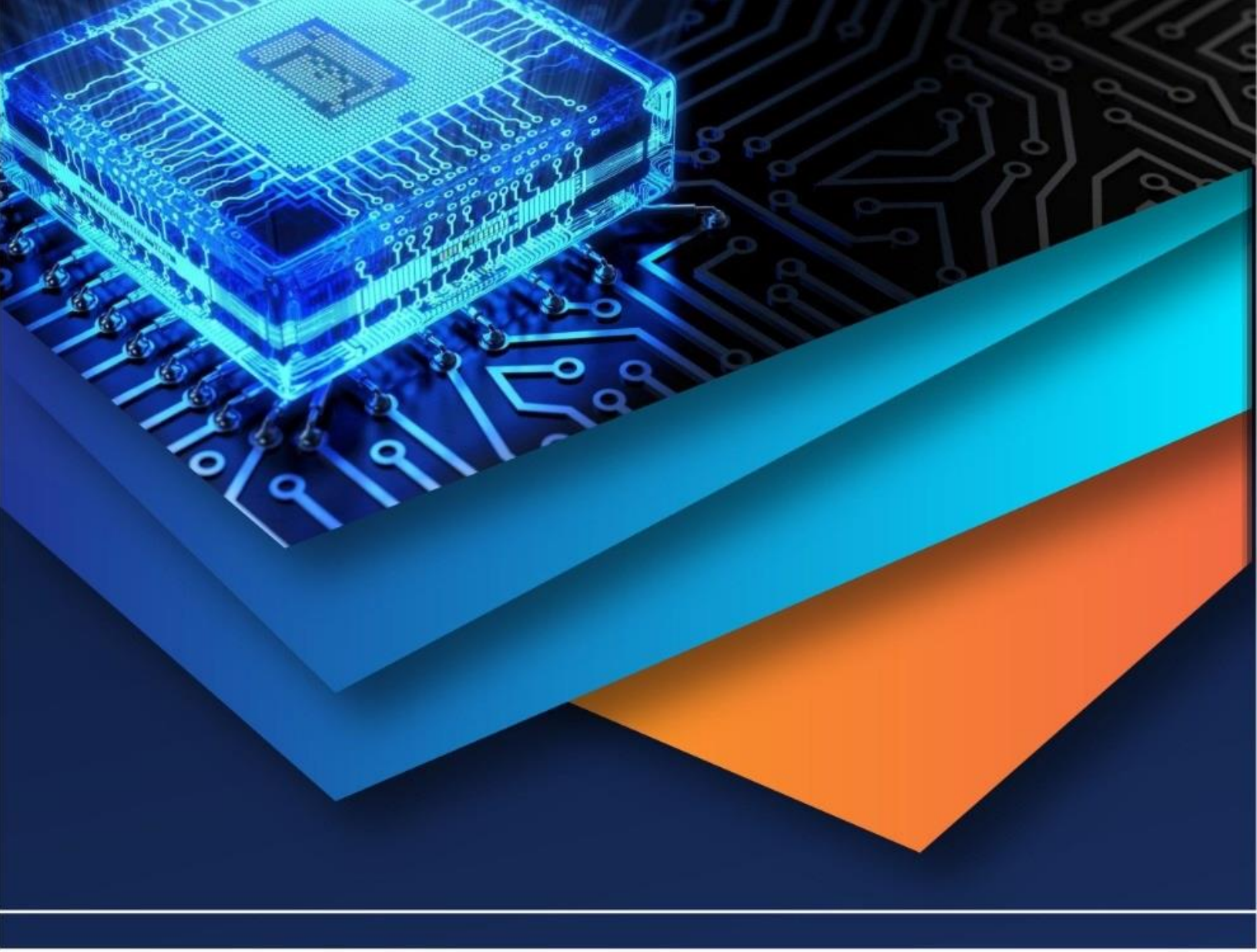
VIII. CONCLUSION AND FUTURE WORK

This paper presented a unified LLM-based smart learning, evaluation, and interview simulation framework using document-grounded prompting to transform static digital education into an interactive, personalized experience. By integrating the Google Gemini API with a full-stack architecture and Cloudinary, the system offered a unified smart learning pipeline covering document summarization, dynamic evaluation, and role-specific interview simulation. Experimental results demonstrated that using context-aware prompting with strict document boundaries significantly reduced AI hallucinations compared to standard ungrounded prompting, achieving over 92% accuracy in test generation and skill-gap detection. Furthermore, the asynchronous Node.js architecture minimized backend latency, meeting the fast processing requirements for real-time educational web applications. The adoption of a modular, secure platform provides a robust solution for students and job seekers facing cognitive overload.

Future work includes OCR support for scanned documents and voice-based interview simulation using speech processing modules.

REFERENCES

[1] P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.
[2] A. Vaswani et al., “Attention Is All You Need,” Advances in Neural Information Processing Systems, vol. 30, 2017.
[3] R. Vatankhah Barenji, N. Salimi, and S. Khoshgoftar, “An LLM-Powered Assessment Retrieval-Augmented Generation for Higher Education,” arXiv preprint arXiv:2601.06141, 2026.
[4] T. Zheng, W. Li, J. Bai, W. Wang, and Y. Song, “Assessing the Robustness of Retrieval-Augmented Generation Systems in K-12 Educational Question Answering,” arXiv preprint arXiv:2412.08985, 2024.
[5] S. Sharma et al., “Retrieval Augmented Generation for Domain-Specific Question Answering,” arXiv preprint arXiv:2404.14760, 2024.
[6] Y. Wang et al., “REAR: A Relevance-Aware Retrieval-Augmented Framework for Open-Domain Question Answering,” arXiv preprint arXiv:2402.17497, 2024.
[7] P. Zhao et al., “Retrieval-Augmented Generation for AI-Generated Content: A Survey,” Data Science and Engineering, 2026.
[8] “Retrieval-Augmented Generation for Educational Application: A Systematic Survey,” Computers and Education: Artificial Intelligence, vol. 8, 2025.
[9] A. Pardasani, K. Maleski, and S. Roy, “Knowledge Retrieval-Based Intelligent Question and Answer Generation Framework for Education,” Journal of Student Research, vol. 14, no. 1, 2025.
[10] B. Aljohani and T. Alsanoosy, “Enhancing Medical Question Answering with LLMs via a Hybrid Retrieval-Augmented Generation Framework,” Information, vol. 17, no. 2, 2026.
[11] P. Chavan and S. Jadhav, “AI Based Mock Interview System,” International Journal of Scientific Research in Science, Engineering and Technology, vol. 13, no. 1, pp. 118–126, 2026.
[12] N. Shrivastava et al., “AI MockPrep: An AI-Driven Interview Simulation and Resume Optimization System,” International Journal of Engineering Research & Technology, vol. 15, no. 1, 2026.
[13] R. B. E. et al., “AI-Powered Resume Analyzer and Interview Preparation System,” International Journal for Research in Applied Science and Engineering Technology, 2025.
[14] V. Verma et al., “AI-Powered Mock Interview System for Automated Skill Assessment,” International Journal for Research in Applied Science and Engineering Technology, 2025.
[15] R. Goli et al., “AI-Powered Mock Interview Preparation,” Zenodo Research Publication, 2025.
[16] J. White et al., “A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT,” arXiv preprint arXiv:2302.11382, 2023.
[17] Google DeepMind, “Gemini: A Family of Highly Capable Multimodal Models,” arXiv preprint arXiv:2312.11805, 2023.
[18] T. Brown et al., “Language Models Are Few-Shot Learners,” Advances in Neural Information Processing Systems, vol. 33, 2020.
[19] Y. Mao et al., “Generation-Augmented Retrieval for Open-Domain Question Answering,” ACL Conference, 2021.
[20] J. Devlin et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” NAACL-HLT, 2019.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY