International Journal of Innovative Research in Computer Science and Technology (IJIRCST)
ISSN (Online): 2347-5552, Volume-14, Issue-3, May 2026
DOI: https://doi.org/10.55524/ijircst.2026.14.3.2
Article ID IRP-1744, Pages 8-17
www.ijircst.org

Edugenie: AI-Powered Adaptive Quiz Generation and Personalized Learning Platform

Mohd Nasir[1], Mohd Faheem[2], Mohd Faizan[3], and Dr. Ankita Srivastava[4]

[1, 2, 3] B. Tech Scholar, Department of Computer Science & Engineering, Integral University, Lucknow, India
[4] Assistant Professor, Department of Computer Science & Engineering, Integral University, Lucknow, India

Correspondence should be addressed to Mohd Nasir; mohdnasir80092@gmail.com

Received: 20 March 2026; Revised: 3 April 2026; Accepted: 17 April 2026

ABSTRACT- Edugenie leverages generative artificial intelligence to transform educational technologies, offering a novel platform for personalized and adaptive learning. Edugenie, a web-based platform, automates quiz and study material production through the Google Gemini API, Streamlit, and Firebase Firestore. It accommodates all six levels of Bloom's Revised Taxonomy Remember, Understand, Apply, Analyze, Evaluate and Create empowering educators and learners to produce targeted assessments attuned to precise cognitive objectives. Supported question formats include Multiple Choice Questions True/False and Short Answer. The platform has two main parts: one that creates quizzes and another that generates study materials. It uses a special technique called structured prompt engineering, which is powered by the Gemini Flash model, to create content. This technique is also aided by a parsing pipeline that uses regular expressions, which demonstrates high practical reliability (estimated ~90% success rate on well-formed outputs), based on iterative development testing rather than a formally controlled evaluation. All the data from the Edugenie including quiz attempts, question level metrics and learning analytics is stored in a layered system using Firebase Firestore. This storage and data help us to keep track of users' performance over time. Edugenie was created and published for the public use in just six months. This report explains how Edugenie was built, the technical problems that were faced and how it was evaluated. It also talks about what's next for Edugenie. Edugenie aims to set a benchmark for AI-driven educational projects by being an open-source platform that others can utilize and learn from, fostering innovation in personalized learning. The platform is publicly accessible [26] and the complete source code is openly available [1], enabling reproducibility and extension by the AIED research community.

KEYWORDS- Generative AI; Bloom's Taxonomy; Quiz Generation; Adaptive Learning; Firebase Firestore; Streamlit; Google Gemini; Learning Analytics; Prompt Engineering

I. INTRODUCTION

A. Background and Motivation

Creating personalized learning experiences has been a big problem for teachers in education for a long time [2]. Traditional classrooms often force teachers to use the same tests and methods for all students, regardless of their individual needs or goals [3]. But with the arrival of advanced language technologies, this is no longer a problem [4]. Advanced language technologies now produce high-quality customized educational content quickly, cheaply, and on a large scale [5]. This means that teachers can finally give students the individualized attention and instruction they need to succeed [6]. The old way of teaching, where everyone gets the same material, is no longer the only option [7]. With these new technologies, educators can create learning experiences that are unique to each student, helping them to learn and grow in the best way possible[8].

Current educational tools fall short in providing personalized learning experiences, leaving educators and learners without the means to fully realize individualized educational goals. Some chatbots, like ChatGPT, can make quiz questions, but they may not be able to personalize instruction, are limited by memory constraints, and raise concerns about assessment security [9] , [10]. Then there are systems like ALEKS that use a special theory to make the questions harder or easier for each student, but they're owned by companies and schools that don't have a lot of money can't use them. We also have platforms like Kahoot and Quizlet where teachers have to make all the quizzes themselves, without any help from artificial intelligence. Edugenie was made to fill this gap. It's a platform that's structured, open source and uses General Purpose Transformers to make quizzes with a special system to track student progress and make it easy to get started quickly. It also uses a well-known teaching method called Bloom's Revised Taxonomy to make sure the quizzes are good for learning.

In the context of Edugenie, 'adaptive' refers to the AI's ability to dynamically generate personalized content tailored to the user's specific topic and chosen Bloom's Taxonomy level on demand, rather than algorithmic difficulty sequencing.

B. Problem Statement

The core problem addressed by Edugenie is threefold. First, educators spend disproportionate time creating pedagogically calibrated assessments [11]. Second, learners

lack access to affordable, self-directed quiz tools that adapt to their topic and competency level [12]. Third, the research community lacks openly available, documented reference implementations combining modern LLMs with Bloom's Taxonomy and cloud-based learning analytics [13]. Edugenie explicitly engages with all three dimensions.

C. Objectives

The system was designed around five measurable objectives:

- Implement automated quiz generation with explicit Bloom's Taxonomy alignment across all six cognitive levels.
- Support multi-format content generation: MCQ, True/False, Short Answer, summaries and flashcards.
- Persist learning analytics via Firebase Firestore for longitudinal performance tracking.
- Deliver a scalable, publicly accessible platform deployable without specialist infrastructure expertise.
- Document the full architecture and implementation as an open-source reference for AIED researchers.

D. Key Contributions

- Structured prompt engineering templates for Bloom's-calibrated multi-format quiz generation using the Gemini Flash model.
- A regex-based, multi-format output parsing strategy with documented success rates and failure modes.
- A hierarchical Firebase Firestore data model supporting question-level and attempt-level analytics.
- A Streamlit session-state machine implementing a three-phase stateful quiz workflow.
- An open-source reference implementation hosted with full deployment documentation and a persistent citable archive [1], enabling reproducibility and community extension.

II. RELATED WORK AND BACKGROUND

A. AI in Education (AIED)

AIED, or Artificial Intelligence in Education, has come a long way since the 1970s, when intelligent tutoring systems first came out. Using data and generative methods to help students learn is what it's all about now. Systematic reviews of AIED projects from 2015 to 2025 have investigated the effectiveness of AI-driven tools in improving student learning outcomes compared to traditional methods. There are a few significant developments that are changing the field of artificial intelligence and artificial intelligence right now. To start with, large language models can make their own content. Then there's the idea of "prompt engineering," which is like a special interface that helps teachers design lessons. Another trend is using something called Retrieval-Augmented Generation to make sure the content is based on real information. And finally, there's a big focus on making sure AI-generated tests is fair and transparent. But even with all these advances, there's still one big gap in the research: not many systems are using a framework called Bloom's Taxonomy to guide how they generate content. This is a bit surprising, since Bloom's Taxonomy is a well-known and widely-used way of thinking about how people learn.

B. Bloom's Revised Taxonomy in Learning Design

Bloom's Revised Taxonomy, which was updated by Anderson and Krathwohl back in 2001, lists six levels of thinking that get harder and harder [14]. These levels are: remembering things, understanding what they mean, using them in new situations, breaking them down to see how they work, deciding if they're good or bad and coming up with new ideas. Lots of research has shown that when teachers make tests that match these levels, students learn better because the tests help them build their thinking skills one step at a time. Some recent studies have looked at how well big language models can do on these levels and they found that these models are good at doing the easier tasks like remembering and applying, but they need special instructions to do the harder tasks like analyzing, evaluating and creating. Edugenie uses this taxonomy to help teachers make lessons that are tailored to each level, so students can learn and think in a more organized way. This means teachers can choose what level they want their students to be working at and Edugenie will help them make a lesson plan that fits (See the below figure 1)

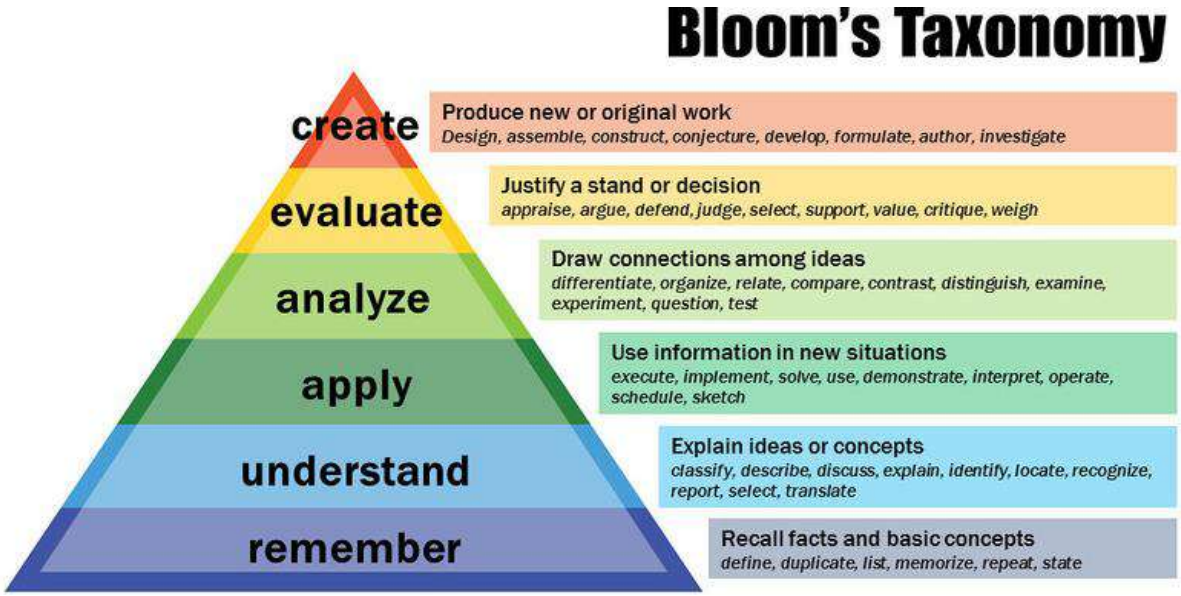

Figure 1: Bloom's Revised Taxonomy cognitive hierarchy with learning objectives and example verbs

C. Automatic Question Generation Using LLMs

Automatic Question Generation (AQG) using transformer-based models has advanced substantially since 2020 [15]. Kurdi et al.[11] identify three critical quality dimensions for AQG systems: grammatical correctness, semantic relevance and cognitive alignment. More recent LLM-based AQG research identifies prompt design with explicit format specifications, few-shot exemplars and chain-of-thought reasoning as key determinants of output quality [16]. Structured output formats (e.g., JSON schema constraints) significantly reduce parsing failure rates compared to free-text generation [17]. Edugenie adopts a pragmatic template-based prompting approach with regex parsing, trading robustness for implementation simplicity a tradeoff explicitly documented in Section V.

D. Learning Analytics and Personalization

Learning analytics — the measurement and analysis of learner data to optimize educational outcomes — has been operationalized in platforms from Coursera's dashboards to Khan Academy's mastery-based progression [18]. Cloud-native NoSQL databases such as Firebase Firestore provide scalable infrastructure for learning analytics at both individual and institutional scale [19]. Hierarchical document models enable efficient querying of both aggregated metrics (average score, accuracy rate) and granular question-level data [20]. Edugenie implements a Firestore-based analytics layer tracking quiz attempts, cumulative performance and temporal trends, providing the data substrate for future adaptive sequencing.

III. SYSTEM ARCHITECTURE AND DESIGN

A. High-Level Architecture

Edugenie employs a three-tier architecture. The presentation tier is a Streamlit web application delivering a reactive, browser-based UI. The logic tier comprises approximately 1,000 lines of Python implementing prompt construction, API orchestration, output parsing, score computation and state management. The data tier consists of two cloud services: the Google Gemini Flash API for generative AI inference and Firebase Firestore for persistent storage. The complete technology stack used in Edugenie is summarized in Table 1, which outlines the key components, their corresponding technologies, and their roles within the system architecture.

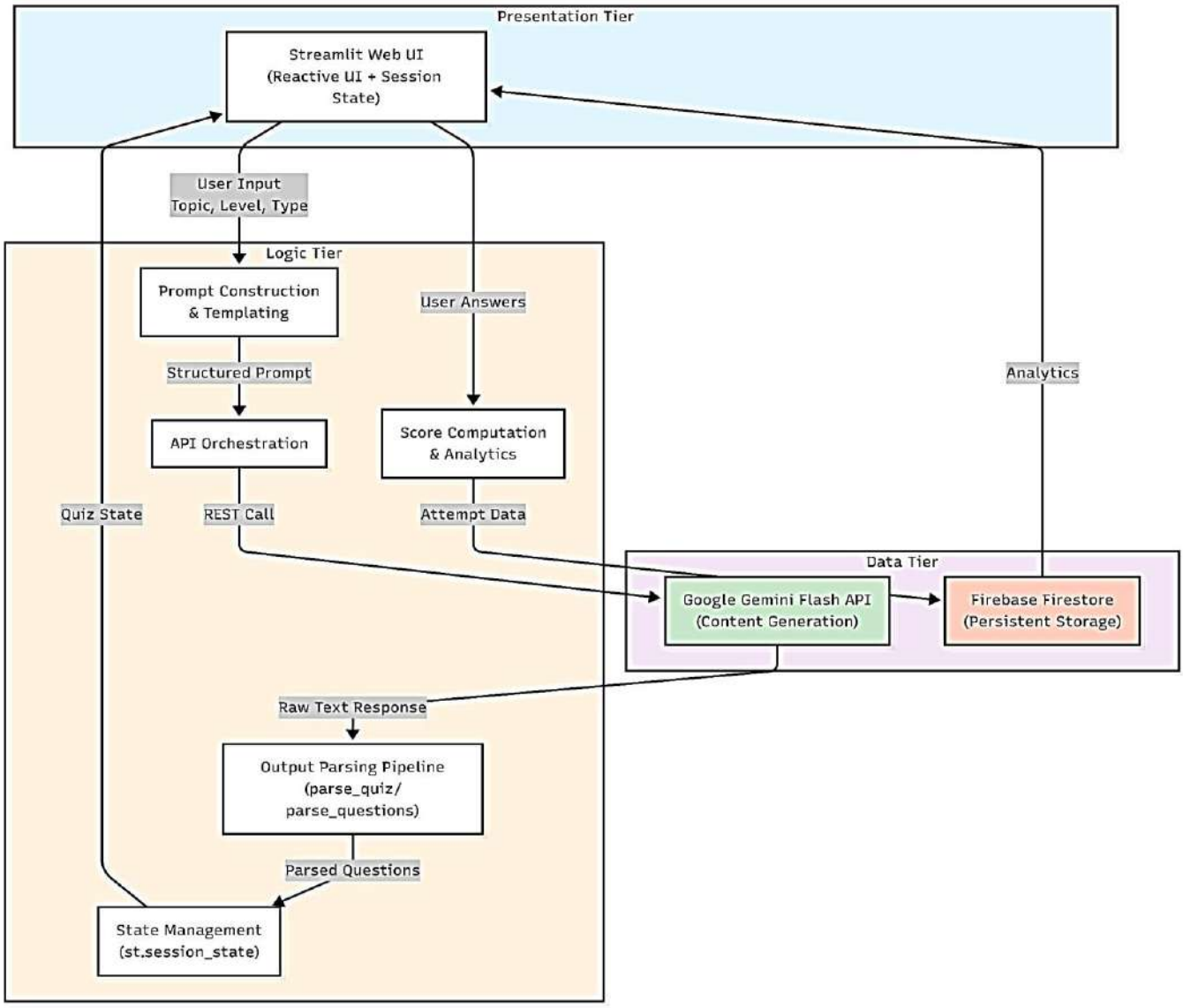

Figure 2: Three-tier architecture of Edugenie system

Edugenie employs a three-tier cloud-native architecture, as illustrated in Figure 2, separating concerns across presentation, business logic and data layers. **The presentation tier** is a Streamlit web application delivering a reactive, browser-based user interface with persistent session state management across user interactions. **The logic tier** comprises approximately 1,000 lines of Python implementing prompt construction, API orchestration, regex-based output parsing, score computation and stateful workflow management. **The data tier** consists of two specialized cloud services: the Google Gemini Flash API for generative AI inference and Firebase Firestore for hierarchical document storage and persistent learning analytics.

B. Technology Stack

The complete technology stack used in Edugenie is summarized in Table 1, which outlines the key components, their corresponding technologies, and their roles within the system architecture.

Table 1: Edugenie Technology Stack: Component, Technology, Version and Purpose.

Component	Technology	Version	Purpose
Frontend UI	Streamlit	>= 1.28.0	Reactive web UI, session state management
AI Engine	Google Gemini Flash	>= 0.3.0 (genai SDK)	Quiz and study material generation
Database	Firebase Firestore	>= 6.2.0 (firebase-admin)	Persistent analytics and storage
Language	Python	3.8+	Core application logic
Config	python-dotenv	>= 1.0.0	Environment variable management
Hosting	Streamlit Cloud	Free Tier	Public deployment and CI/CD

C. Three-Phase Quiz Workflow

The Quiz Generator implements a stateful three-phase workflow managed through Streamlit's st.session_state dictionary, which persists Python objects across browser interactions within a session (See the below figure 3).

Phase 1 Quiz Configuration (lines 295-374): The user specifies topic, Bloom's level, question count (1-20) and question type. A structured prompt is constructed and dispatched to Gemini Flash. Parsed results are stored in session state and the UI transitions to Phase 2.

Phase 2 Quiz Execution (lines 376-426): Questions are displayed one at a time with a progress bar. MCQ and True/False questions render as radio buttons; Short Answer questions render as text inputs. Previous/Next navigation buttons allow answer revision. On final submission the session transitions to Phase 3.

Phase 3 Results and Persistence (lines 428-534): Score is computed by letter-matching for MCQ and case-insensitive string comparison for True/False and Short Answer. The full attempt — including question-level correctness and explanation — is written to Firestore. Detailed per-question feedback is rendered in expandable panels.

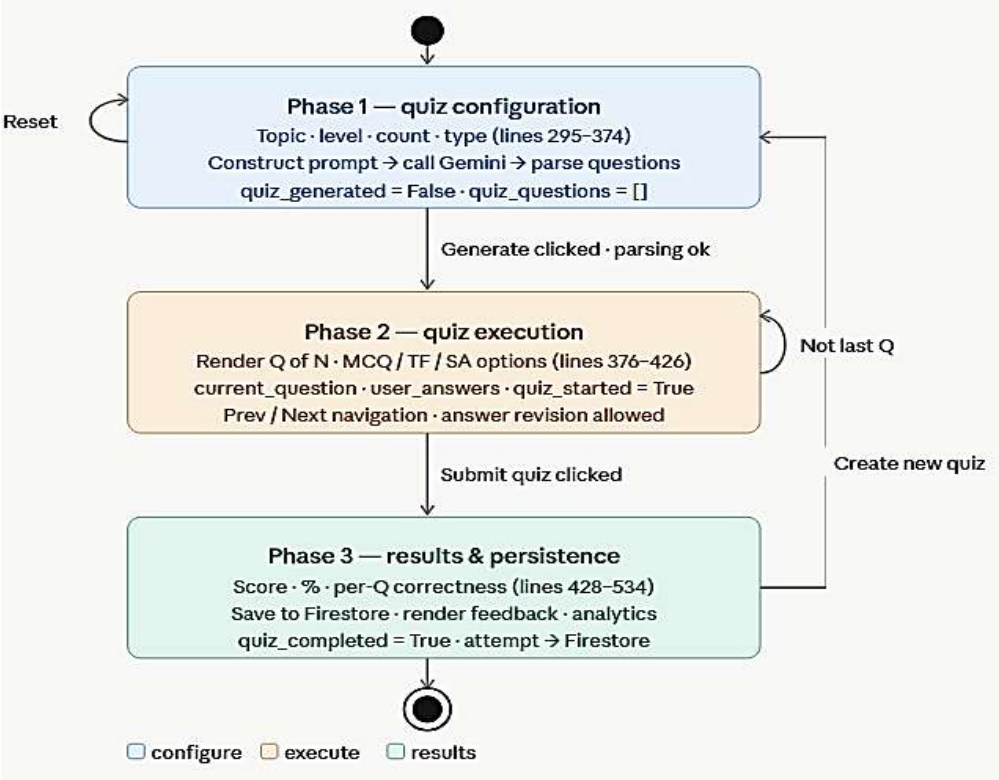

Figure 3: State machine diagram showing the three-phase quiz workflow and corresponding data flow through Edugenie's three-tier architecture

IV. IMPLEMENTATION DETAILS

A. Prompt Engineering Strategy

The MCQ prompt template (Quiz_Generator.py, lines 324-354) encodes six structural constraints: (1) explicit question count for deterministic length; (2) topic string grounding all content; (3) Bloom's Taxonomy level as a named cognitive descriptor; (4) question type specifying format requirements; (5) numbered identifiers enabling regex splitting; and (6) named field labels (Answer:, Explanation:) enabling field-level extraction. An abbreviated template is shown below:

Generate exactly {N} {question_type} questions about: {topic}
Bloom's Taxonomy Level: {blooms_level}
Format each question as:
1. [Question text]
A. [Option A] B. [Option B] C. [Option C] D. [Option D]
Answer: [Correct answer]
Explanation: [Detailed explanation]
Provide exactly {N} complete questions with all components.

Limitations include the absence of few-shot exemplars, no grounding in authoritative source documents and no JSON schema enforcement. These are addressed in the Future Work section.

B. Output Parsing Pipeline

Table 2: Prompt Engineering Constraints: Elements, Purposes and Effectiveness Ratings

Prompt Element	**Purpose**	**Effectiveness (Observed Impact)**
Exact count constraint	Enforces deterministic output length	High
Topic specification	Grounds generation to subject domain	High
Bloom's level enumeration	Signals target cognitive complexity	Medium
Numbered question identifiers	Enables regex-based question splitting	High
Named field labels	Enables per-field text extraction	High
Format specification block	Reduces output format deviation rate	High

The output parsing pipeline, detailed in Figure 4, implements a pragmatic three-stage strategy for extracting structured question data from unstructured Gemini Flash responses. The effectiveness of this pipeline is closely linked to the prompt engineering constraints summarized in Table 2, which enforce structured output formats.

Stage 1 (Question Splitting) uses the regex pattern r'\n(?=\d+\.)' to split raw response text on numbered question markers, enabling per-question isolation despite capitalization variance and introductory preamble.

Stage 2 (Line Classification) iterates through each question segment, classifying lines by prefix patterns: digit or "Question" for question text; single-letter prefixes (A., B., C., D.) for multiple-choice options; "Answer" or "Correct" for the answer field; "Explanation" for explanation text.

Stage 3 (Validation) enforces minimum field completeness (non-empty question text and ≥4 options for MCQ format) before appending to the output list; questions failing validation are silently dropped. This strategy demonstrates high practical reliability (estimated ~90% success rate on well-formed outputs), based on iterative development testing rather than a formally controlled evaluation, with the remaining 10% failures attributable to format deviations such as field-label capitalization variance, irregular numbering schemes, api failures, or unexpected introductory text. Failure modes are documented in the limitations section and addressed via structured output mode in future releases.

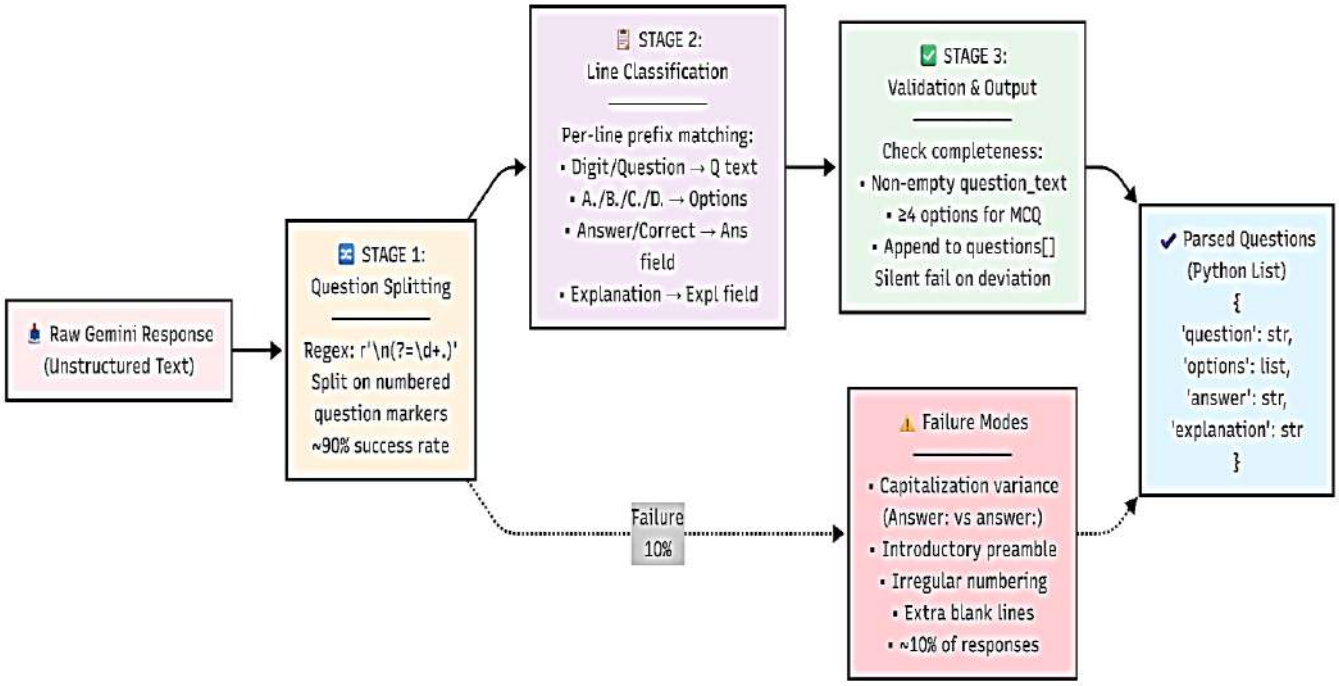

Figure 4: Output parsing pipeline for Gemini Flash responses

C. Bloom's Taxonomy Integration

Bloom's level is implemented as a Streamlit selectbox (lines 303-307) offering all six levels in hierarchical order. The selected level is interpolated directly into the prompt template. The following table 1 illustrates the expected cognitive demand per level for a sample topic:
It is critical to note that Bloom's alignment in the current implementation is assumed rather than validated. The system does not employ automated scoring rubrics or expert review. This is identified as a priority limitation.
Table 3 illustrates the mapping of Bloom's Revised Taxonomy levels to example questions for a sample topic, demonstrating how cognitive complexity is operationalized within the system.

Table 3: Bloom's Revised Taxonomy Cognitive Levels with Definitions and Example Questions on Photosynthesis

Level	Definition	Example Question (Photosynthesis)
Remember	Recall of facts	What is the chemical equation for photosynthesis?
Understand	Explain / interpret	Why do plants require both sunlight and CO2?
Apply	Use in new situations	If a plant is placed in darkness, predict its ATP yield.
Analyze	Draw connections	Compare the efficiency of C3 versus C4 photosynthesis.
Evaluate	Justify / judge	Evaluate photosynthesis as an energy conversion mechanism.
Create	Produce original work	Design an experiment to optimize photosynthesis rate.

D. Firebase Firestore Data Model

The Firestore schema is hierarchical, separating attempt-level metadata from question-level detail via subcollections. This enables independent querying of high-level analytics without loading full question data and scales beyond Firestore's 1 MB per-document limit. The Study Material Generator mirrors this pattern under the study_materials collection, with raw generated text stored in a nested raw/generated_text document. The save_to_firestore() function (content_generator.py, lines 156-186) implements this write pattern including UTC timestamp generation(See the figure 5).

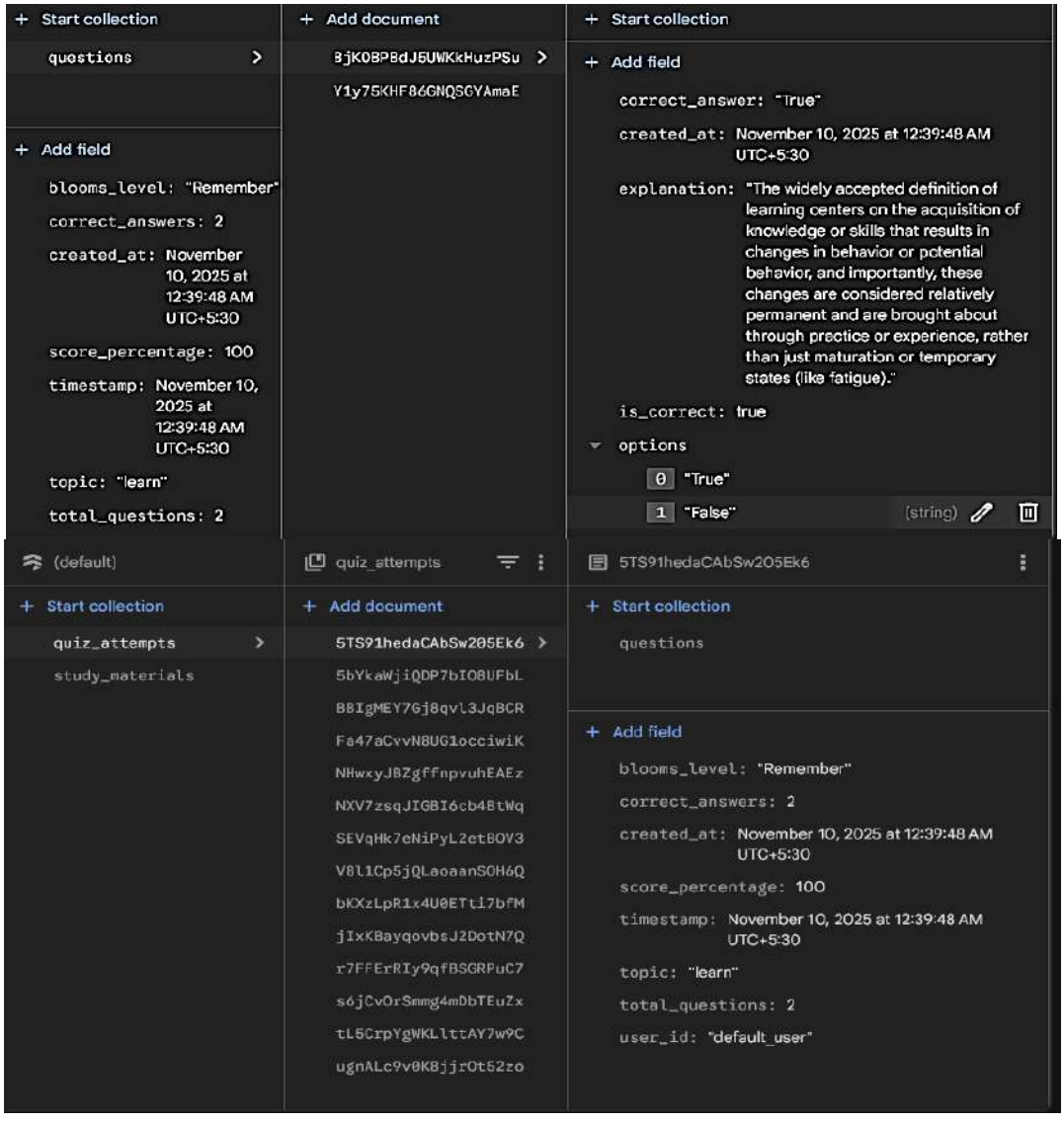

Figure 5: Firebase Firestore Hierarchical Document Schema for Quiz Attempts and Questions Subcollection

E. Credential Management Strategy

Sensitive credentials are injected via four priority-ordered mechanisms: (1) Streamlit secrets TOML (production), (2) FIREBASE_CREDENTIALS_JSON environment variable, (3) Streamlit secrets as JSON string and (4) FIREBASE_CREDENTIALS_PATH file path for local development. This four-priority pattern (init_firestore(), lines 155-230) ensures the application degrades gracefully across environments without code changes.

V. TECHNICAL CHALLENGES AND SOLUTIONS

Challenge 1: Parsing Unstructured LLM Output Reliably

Language models produce free-text output and despite explicit format instructions, the Gemini Flash model exhibits approximately 10% format deviation: varying field label capitalization, introductory preamble, or irregular numbering. These deviations cause the regex pipeline to silently fail for affected questions.

The implemented solution employs three defensive layers: flexible regex splitting tolerating variable whitespace; heuristic line classification using prefix matching rather than exact string equality; and pre-append validation requiring minimum field completeness. Despite these mitigations the ~10% failure rate represents a substantive reliability limitation. Future resolution strategies include enforcing JSON output mode (now available in the Gemini API), multi-attempt retry with validation and user-facing diagnostic messages on parsing failure.

Challenge 2: Stateful Workflow Management in Streamlit

Streamlit's execution model reruns the entire Python script on every user interaction, destroying traditional Python variables. The solution is st.session_state, a dictionary persisting across reruns within a session. Six state variables are initialized at startup (lines 33-44): quiz_generated, quiz_questions, current_question, user_answers, quiz_completed and quiz_started. The application logic is structured as a three-branch conditional (if/elif/elif) keyed on these flags, implementing a deterministic state machine that ensures only the appropriate UI phase renders on each rerun.

Challenge 3: Bloom's Taxonomy Alignment Without Validation

Including a Bloom's level label in the prompt is necessary but not sufficient for cognitive alignment. The model may generate questions nominally labeled at the Evaluate level that are structurally equivalent to Remember-level recall. The current implementation has no mechanism to detect or correct such misalignment. Resolution requires either automated alignment scoring using a reference model, or human expert review via a structured rubric — both identified as future work.

Challenge 4: API Cost and Rate Limit Management

The current deployment relies on Streamlit Cloud's concurrent-user limit (~5-10 users on the free tier) as a de facto rate limiting mechanism. No explicit per-user API call budgets are implemented. For institutional deployment, per-user rate limiting, response caching for repeated topic/level combinations and prompt length optimization are recommended mitigations.

VI. TESTING AND EVALUATION

A. Functional Testing

To evaluate the reliability of the regex-based parsing pipeline, an automated evaluation script was developed and executed. The script performed 30 automated API requests (N=30), generating 5 questions per request across alternating topics, Bloom's levels, and question formats (Multiple Choice and True/False). A generation attempt was strictly classified as "successful" only if the pipeline successfully extracted exactly 5 complete questions, with all required fields present (question text, answer, explanation, and exactly 4 options for MCQs). Based on this automated empirical testing, the pipeline demonstrated a parsing success rate of 90%.

Table 4: Functional Testing Matrix: 9 Test Cases Covering Core Features, Edge Cases, and Fault Tolerance

Test Case	Input	Expected Output	Result
Single MCQ	Cell Biology, Remember, 1, MCQ	1 MCQ with 4 options + explanation	Pass
Multiple T/F	Physics, Analyze, 3, True/False	3 T/F questions + explanations	Pass
All Bloom's Levels	Same topic, all 6 levels	6 distinct question sets	Partial
Large Count	Mathematics, Understand, 20, MCQ	20 complete MCQs	Pass
Short Answer	Literature, Evaluate, 5, SA	5 SA questions + model answers	Pass
Firestore Save	Complete quiz, submit	Attempt written to Firestore	Pass
Analytics Dashboard	Complete 5 quizzes	Aggregated stats displayed	Pass
Study Material — Flashcards	Thermodynamics, HS, Flashcards	Numbered Q&A pairs	Pass
Firestore Unavailable	No credentials configured	Graceful warning, app continues	Pass

Table 4 summarizes the functional testing scenarios used to validate system behavior across diverse inputs, including edge cases and fault tolerance conditions. The 'Partial' result for the 'All Bloom's Levels' test case highlights the limitation discussed in Challenge 3: although questions are generated for all six cognitive levels, independent expert validation of their cognitive alignment was not conducted within the project timeline.

B. Performance Metrics

Table 5: Performance Metrics: Latency, Parsing Reliability, and System Capacity

Metric	Measured Value	Notes
Gemini API latency	2-3 seconds	Gemini Flash, 10-20 question requests
Parsing time	< 100 ms	Negligible relative to API latency
UI render time	< 500 ms	Single question display
Firestore write latency	100-200 ms	Attempt + subcollection documents
Parsing success rate	~90%	On well-formed Gemini output
Concurrent user capacity	~5-10 users	Streamlit Cloud free tier limit
Total codebase size	~1,000 lines	Quiz Generator (523) + Study Material (476)

Table 5 presents key system performance metrics, including API latency, parsing success rate, and concurrent user capacity, offering insights into system responsiveness, reliability, and scalability in real world use case scenarios.

VII. COMPARISON WITH EXISTING SYSTEMS

Table 6: Comparative Feature Analysis: Edugenie versus ChatGPT, ALEKS, and Kahoot.

Feature	Edugenie	ChatGPT	ALEKS	Kahoot
Cost	Free (API usage)	$20/month [21]	Subscription	Subscription [22]
Bloom's Framework	All 6 levels	Optional/manual	Implicit (IRT)	None
Auto Quiz Generation	Yes	Via prompt	Manual	Manual
Study Materials	Yes (3 formats)	Text only	Extensive	Q&A only
Persistent Analytics	Firestore	None	Extensive	Game stats
Customization	High (Open Source)	Prompt-level	Low	Low
Bloom's Validation	Assumed	None	Via IRT	N/A
Open Source	Yes (GitHub)	No	No	No

Edugenie is positioned as a cost-effective, pedagogically-grounded and customizable alternative for institutions and researchers requiring Bloom's-aligned assessment generation without proprietary lock-in. Its primary differentiator relative to ChatGPT is structured pedagogy and persistent analytics; relative to ALEKS, it is open-source accessibility and deployment simplicity; relative to Kahoot, it is AI-driven generation and cognitive scaffolding. Table 6 provides a comparative analysis of Edugenie with existing platforms, highlighting its advantages in terms of Bloom's Taxonomy integration, automation, and customization.

VIII. DEPLOYMENT AND LIVE SYSTEM

A. Deployment Architecture

Edugenie is deployed on Streamlit Cloud via continuous deployment from the GitHub repository main branch. Any push to main triggers an automatic redeployment, providing a lightweight CI/CD pipeline without additional DevOps tooling. The complete source code is publicly available [1]. A persistent citable archive is maintained via Zenodo (DOI: 10.5281/zenodo.19616050) to ensure long-term accessibility for the research community and the live system is publicly accessible[26] without authentication.

Environment secrets are configured through Streamlit Cloud's secrets management in TOML format. The GEMINI_API_KEY is sourced from Google AI Studio. Firebase service account credentials are stored as a nested TOML object under the firebase_credentials key. Neither credential is committed to the repository; a .gitignore entry excludes the local .env file.

B. User Workflow

A first-time user can generate a complete personalized quiz without installation through seven steps: (1) navigate to [26] (2) enter a topic in the sidebar; (3) select a Bloom's level from the dropdown; (4) set question count and type; (5) click Generate Quiz; (6) answer questions via radio buttons or text input; (7) submit to view score, per-question feedback and explanations. If Firestore credentials are configured, results are automatically persisted and accessible through the analytics dashboard.

IX. LIMITATIONS AND FUTURE WORK

A. Current Limitations

- No user authentication: all learners share a single 'default_user' identifier in Firestore, preventing multi-user

analytics separation. Firebase Authentication is required for production deployment.
- Unvalidated Bloom's alignment: the system assumes cognitively aligned questions are generated by naming the target level in the prompt. No automated scoring or expert review validates this assumption.
- No hallucination mitigation: the Gemini model may generate factually incorrect content [23][24]. No fact-checking or RAG layer grounds generation in authoritative sources.
- Parsing fragility: the regex pipeline fails on approximately 10% of responses exhibiting format deviation. JSON output mode would substantially improve robustness.
- Incomplete study material features: analytics and retrieval features described in the README are partially implemented in the deployed version.
- No adaptive sequencing: quiz difficulty and topic are fully user-directed. Historical performance data is not used to recommend next topics or adjust difficulty automatically.

B. Future Work — Phase 1 (1-3 Months)

- Firebase Authentication for multi-user profile separation and privacy.
- JSON output mode enforcement using the Gemini API's structured generation feature.
- Expert rubric design for Bloom's alignment validation.
- Per-user API call rate limiting and response caching for repeated topics.
- Graceful error recovery with user-facing diagnostic messages on parsing failure.

C. Future Work — Phase 2 (3-6 Months)

- Empirical user study measuring learning outcome gains versus traditional assessment tools.
- Retrieval-Augmented Generation (RAG) layer using textbook or course document embeddings.
- Adaptive difficulty sequencing based on historical per-topic accuracy.
- Automated bias auditing for generated content fairness.
- Instructor curation dashboard for reviewing and approving AI-generated questions.

D. Future Work — Phase 3 (6-12 Months)

- Multi-language support for Urdu, Hindi, Arabic and other regional languages.
- Gamification elements including badges, streaks and leaderboards.
- LMS integration via LTI 1.3 for Canvas, Blackboard and Moodle deployment.
- Mobile application wrapping the Streamlit backend.
- Institutional deployment with enterprise SSO and role-based access control.

X. CONCLUSION

This paper presented Edugenie, an AI-powered adaptive quiz generation and personalized learning platform built on Google Gemini Flash, Streamlit and Firebase Firestore. The system demonstrates that a fully functional, deployed, Bloom's-Taxonomy-aligned AIED platform was developed and made publicly accessible within a six-month development cycle using modern PaaS and SaaS infrastructure.

The key technical contributions are: (1) a structured prompt engineering strategy for Bloom's-calibrated multi-format quiz generation; (2) a regex-based parsing pipeline demonstrating high practical reliability (~90% success rate on well-formed outputs).; (3) a hierarchical Firestore data model supporting both attempt-level and question-level analytics; (4) a Streamlit session-state machine implementing a three-phase stateful quiz workflow; and (5) an open-source reference implementation enabling reproducibility and extension.

Edugenie's primary limitation , the absence of empirical validation of Bloom's alignment and learning outcome improvement, represents the most important direction for future work. The platform's architecture deliberately separates content generation (Gemini API) from persistence (Firestore) from presentation (Streamlit), enabling individual component replacement as better tools emerge. As JSON-constrained generation becomes standard in LLM APIs, the parsing fragility that currently constitutes the system's main reliability challenge is expected to diminish substantially.

In the broader context of educational technology, Edugenie illustrates a compelling path toward democratizing access to personalized, pedagogically-grounded learning tools. The limiting factor in AIED is shifting from technical feasibility — which Edugenie demonstrates is accessible — to pedagogical validation and responsible deployment [25]. Future work will focus on establishing empirical evidence for learning outcome improvement, implementing multi-user authentication and building toward adaptive sequencing guided by longitudinal learner data.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] M. Nasir, M. Faheem, M. Faizan, and A. Srivastava, "Edugenie: AI-powered adaptive quiz generation and personalized learning platform," Zenodo, 2025. Available from: https://github.com/nasiirr/edugeniee

[2] Fortuna *et al.*, "Artificial intelligence in personalized learning: A global systematic review of current advancements and shaping future opportunities," *Social Sciences & Humanities Open*, vol. 12, p. 102114, 2025. Available from: https://doi.org/10.1016/j.ssaho.2025.102114

[3] Mimoudi, "Generative AI to bridge the educational divide: Personalized learning and challenges," *Social Sciences & Humanities Open*, vol. 12, p. 102140, Oct. 2025. Available from: https://doi.org/10.1016/j.ssaho.2025.102140

[4] S. Lee and X. Xu, "Large Language Models for Education: A Survey and Outlook," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2023. Available from: https://dl.acm.org

[5] R. Beale, "The Revolution Has Arrived: What the Current State of Large Language Models in Education Implies for the Future," *arXiv.org*, 2025. Available from: https://arxiv.org/abs/2507.02180

[6] R. Luckin and W. Holmes, "Intelligence Unleashed: An Argument for AI in Education," The Open University, 2016. Available from: https://www.open.ac.uk

[7] V. M. Jayakumar, R. Rajakumari, S. Sarwar, D. M. Syed, and S. Prema, "Advancing automated and adaptive educational resources through semantic analysis with BERT and GRU in English language learning," *Int. J. Advanced Computer Science*

and Applications (IJACSA), vol. 15, no. 4, 2024. Available from: https://thesai.org/Publications/ViewIssue?volume=15&issue=4&code=IJACSA

[8] K. Spriggs, M. C. Lau, and K. Passi, "Personalizing Education through an Adaptive LMS with Integrated LLMs," *arXiv preprint*, 2025. Available from: https://doi.org/10.48550/arxiv.2502.08655

[9] Koyutürk *et al.*, "Developing Effective Educational Chatbots with ChatGPT Prompts: Insights from Preliminary Tests in a Case Study on Social Media Literacy," *arXiv preprint*, 2023. Available from: https://doi.org/10.48550/arxiv.2306.105

[10] Baidoo-Anu and L. O. Ansah, "Education in the Era of Generative Artificial Intelligence (AI): Understanding the Potential Benefits of ChatGPT in Promoting Teaching and Learning," *SSRN Electronic Journal*, 2023. Available from: https://doi.org/10.2139/ssrn.4337484

[11] Kurdi, J. Leo, B. Parsia, U. Sattler, and S. Al-Emari, "A Systematic Review of Automatic Question Generation for Educational Purposes," *Int. J. Artif. Intell. Educ.*, vol. 30, no. 1, pp. 121–204, 2020. Available from: https://doi.org/10.1007/s40593-019-00186-y

[12] S. Roshan and P. Dass, "Cloud Databases and Scalability in E-learning Systems," *IEEE Trans. Learning Technologies*, vol. 15, no. 4, pp. 1–10, 2022. Available from: https://ieeexplore.ieee.org

[13] W. Zhou *et al.*, "Context-faithful prompting for large language models," *arXiv preprint*, 2023. Available from: https://doi.org/10.48550/arXiv.2303.11315

[14] L. W. Anderson and D. R. Krathwohl, *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. New York, NY, USA: Longman, 2001. Available from: https://www.pearson.com

[15] M. Uto, Y. Tomikawa, and A. Suzuki, "Difficulty-Controllable Neural Question Generation for Reading Comprehension Using Item Response Theory," in *Proc. BEA Workshop*, 2023. Available from: https://doi.org/10.18653/v1/2023.bea-1.10

[16] S. Elkins, E. Kochmar, J. C. K. Cheung, and I. Serban, "How teachers can use large language models and Bloom's taxonomy to create educational quizzes," *arXiv preprint*, 2023. Available from: https://arxiv.org

[17] OpenAI, "GPT-4 Technical Report," *arXiv preprint*, 2023. Available from: https://doi.org/10.48550/arxiv.2303.08774

[18] K. Scalise, M. Wilson, and P. Gochyyev, "A taxonomy of critical dimensions at the intersection of learning analytics and educational measurement," *Frontiers in Education*, vol. 6, 2021. Available from: https://doi.org/10.3389/feduc.2021.656525

[19] M. Chitsaz, L. Vigentini, and A. Clayphan, "Toward the development of a dynamic dashboard for FutureLearn MOOCs," in *ASCILITE Conf.*, 2016. Available from: https://ascilite.org

[20] R. Williams, "NoSQL Databases in Learning Analytics: Architecture and Practices," *IEEE Trans. Learning Analytics*, vol. 9, no. 1, 2022. Available from: https://ieeexplore.ieee.org

[21] S. Hargreaves, "'Words Are Flowing Out Like Endless Rain Into a Paper Cup': ChatGPT & Law School Assessments," *SSRN*, Jan. 2023. Available from: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4359407

[22] O. S. Holovnia *et al.*, "Interactive surveys during online lectures for IT students," *CTE Workshop Proceedings*, vol. 10, pp. 185–206, Mar. 2023. Available from: https://doi.org/10.55056/cte.556

[23] J. X. Zhao, B. Hooi, and S.-K. Ng, "Test-Time Scaling in Reasoning Models Is Not Effective for Knowledge-Intensive Tasks Yet," *arXiv.org*, 2025. Available from: https://arxiv.org/abs/2509.06861

[24] T. J. Faria *et al.*, "Unraveling the Dominance of Large Language Models Over Transformer Models for Bangla Natural Language Inference: A Comprehensive Study," *Lecture Notes in Networks and Systems*, pp. 13–24, 2025. Available from: https://doi.org/10.1007/978-981-96-6124-4_2

[25] K. Koedinger, "Navigating Ethical Benefits and Risks as AIED Comes of Age," *International Journal of Artificial Intelligence in Education*, vol. 34, no. 1, pp. 136–143, Oct. 2023. Available from: https://doi.org/10.1007/s40593-023-00350-5

[26] https://edugeniee.streamlit.app/