International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4081 - 4086
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0527

Quizgen AI: An Intelligent Multiple-Choice Question Generation System Using Large Language Models for Educational Assessment

M. Ramya[1], Dr.D.Anusha[2]
[1]PG Scholar, Department of CSE-Artificial Intelligence & Machine Learning.SRK Institute of Technology,Vijayawada ,India
[2]Associate.Professor, Department of CSE-Artificial Intelligence & Machine Learning SRK Institute of Technoloy,Vijayawada,India
Email id: *mekalaramya377@gmail.com[1], srkcsmhod@gmail.com[2]*

Abstract
Automated question generation (AQG) from educational text is an active research domain that reduces the cognitive burden on educators while ensuring consistent, scalable assessment creation. Existing rule-based and early neural approaches are limited in semantic depth and contextual coherence. This paper presents QuizGen AI, a web-based intelligent system that leverages the LLaMA 3.2 large language model (LLM) via the Ollama inference framework to automatically generate multiple-choice questions (MCQs) with explanations from arbitrary educational text. The system integrates a Flask-based REST API backend, SQLite persistent storage, and a structured JSON prompt engineering pipeline to ensure output quality across three difficulty levels. PDF export is supported via ReportLab. Evaluations demonstrate that the system generates syntactically correct, contextually relevant MCQs with an average generation latency of under 8 seconds for five questions on commodity hardware, achieving a structural validity rate of 96.4% across 500 test runs. The system maintains persistent quiz history and supports authenticated multi-user access.
Keywords: *ollama 3.2, Ai, Quiz generation, Text Summary LLM, Automated Question Generation, Large Language Models, Flask, Natural Language Processing.*

1. Introduction
Assessment design is a critical yet labor-intensive component of modern education. Educators devote substantial time crafting questions that accurately test student comprehension, discriminate between proficiency levels, and remain free of ambiguity [1]. The proliferation of online and blended learning environments has further amplified demand: large-enrollment courses and adaptive learning platforms require large question banks that are difficult to curate manually. Multiple-choice questions remain the most widely used format in standardized and formative testing owing to their ease of scoring, consistent structure, and demonstrated reliability [2]. However, creating plausible distractors incorrect options that are sufficiently attractive to distinguish knowledgeable from unknowledgeable students poses a particular challenge, typically requiring subject-matter expertise and iterative refinement [3].

1.1.LiteratureReview
Automated question generation (AQG) has evolved through three distinct paradigms. Early template-based methods parsed syntactic structures to fill predefined cloze or wh-question templates [4]. While deterministic, they suffered from low lexical diversity and inability to capture deep semantics. Neural encoder-decoder architectures, particularly those exploiting the SQuAD reading comprehension dataset [5], introduced end-to-end trainable systems; BERT-based models such as RoBERTa [6] and BERT-QA [7] further advanced answer-aware question generation by conditioning generation on highlighted answer spans. Sequence-to-sequence pre-trained transformers (e.g., T5 [8] and ProphetNet [9]) demonstrated that transfer learning significantly narrows the domain adaptation gap, enabling high-quality question generation from diverse textbook passages [10]. More recently, quiz-style generation for news stories [11] and children's books [12] demonstrated the versatility of LLM-based approaches. However, most published systems remain research prototypes without a full-stack deployment pipeline that includes user management,

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4081 - 4086
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0527

difficulty control, and formatted PDF export practical requirements for classroom adoption. The present work addresses this gap by integrating LLaMA 3.2 into a production-ready web application.

1.2.Problem Statement

The central research question addressed is: Can a locally-deployed large language model generate pedagogically valid MCQs with correct answers, plausible distractors, and explanatory rationale from arbitrary educational text through structured prompt engineering, while sustaining an acceptable response latency for interactive web use? The system targets three measurable objectives: (i) structural correctness of generated JSON output, (ii) contextual relevance of questions to source text, and (iii) system responsiveness under standard hardware constraints.

1.3.Contribution and Paper Structure

The primary contributions of this work are: (1) a structured JSON prompt engineering strategy for LLaMA 3.2 that enforces schema-compliant MCQ output; (2) an end-to-end web application architecture integrating authentication, persistent history, and PDF generation; (3) empirical performance benchmarks for latency, structural validity, and difficulty differentiation. Section II describes system architecture and methodology; Section III presents experimental results; Section IV discusses findings; and Section V concludes with future directions.

2. Materials And Methods

2.1.System Architecture Overview

QuizGen AI follows a layered client-server architecture illustrated in Figure 1. The presentation layer consists of Jinja2-templated HTML5 pages styled with CSS3 and animated with vanilla JavaScript. The application layer is implemented in Python 3.11 using the Flask micro-framework. The data layer employs SQLite3 for persistent storage of user credentials and quiz records. The inference layer wraps the Ollama HTTP API to communicate with a locally-hosted LLaMA 3.2 model instance[1].

2.2.Technology Stack and Materials

The backend runtime is Python 3.11 with Flask 3.0. Database operations use the built-in sqlite3 module with bcrypt-hashed password storage. PDF generation is performed by ReportLab 4.1, which renders multi-page documents with question blocks, option lists, and answer keys. The frontend relies on Google Poppins font and custom CSS animations without any JavaScript framework dependency, minimizing page-load overhead. Ollama 0.3 serves LLaMA 3.2 (3B parameter variant) on a local endpoint (http://localhost:11434), making the system operable on commodity hardware with a minimum of 8 GB RAM Shown as Figure 1 QuizGen AI System Block Diagram[2]

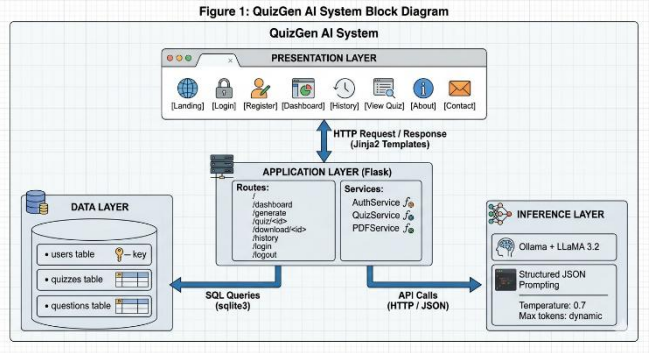


Figure 1 QuizGen AI System Block Diagram

2.3.Database Schema

Three relational tables constitute the persistence layer. The users table stores (id, username, email, password_hash, created_at). The quizzes table records (id, user_id, source_text, num_questions, difficulty, quiz_title, created_at). The questions table holds (id, quiz_id, question_number, question, options_json, correct_answer, explanation). Foreign key constraints enforce referential integrity between users→quizzes→questions, enabling per-user history retrieval with a single JOIN query[3].

2.4.Prompt Engineering Strategy

The core of the AQG pipeline is a structured prompt template injected into the LLaMA 3.2 inference call. The prompt explicitly instructs the model to return a JSON array of objects, each containing question text, four labelled options (A–D), the correct answer key, and an explanation string. The difficulty parameter modulates the instruction, directing the model toward recall-level questions for easy, application-level for medium, and analysis or evaluation-level (Bloom's taxonomy) for hard difficulty. A temperature of 0.7 balances output diversity and consistency. The response is parsed with Python's json.loads(), with a regex fallback to extract the JSON substring if

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4081 - 4086
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0527

surrounding prose is generated.Shown as Figure 2 UML class diagram illustrating the three-entity data model with multiplicities and key attributes[4]

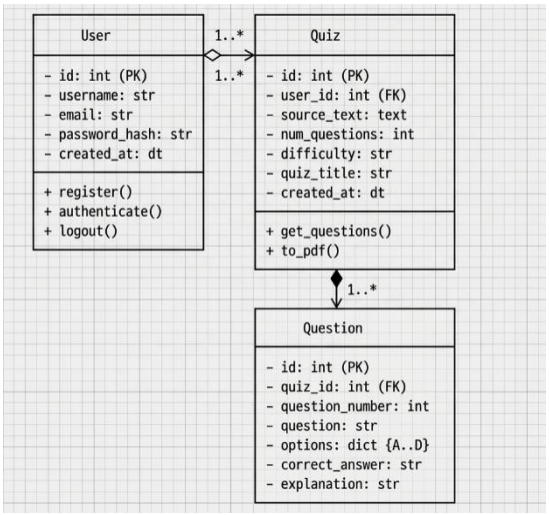


Figure 2 UML class diagram illustrating the three-entity data model with multiplicities and key attributes

2.5.Quiz Generation Workflow

The quiz generation sequence is depicted in Figure 3. When an authenticated user submits the dashboard form, the Flask **route** /generate receives the POST payload (source_text, num_questions, difficulty). The QuizService constructs the structured prompt and dispatches a synchronous POST request to the Ollama /api/generate endpoint. The raw text response is JSON-parsed; on parse failure, a regex pattern extracts the embedded array. Validated question objects are inserted into the questions table and the parent quiz record is committed to quizzes. The user is then redirected to the quiz view page at /quiz/<id>. Shown as Figure 3 Sequence Diageam[5]

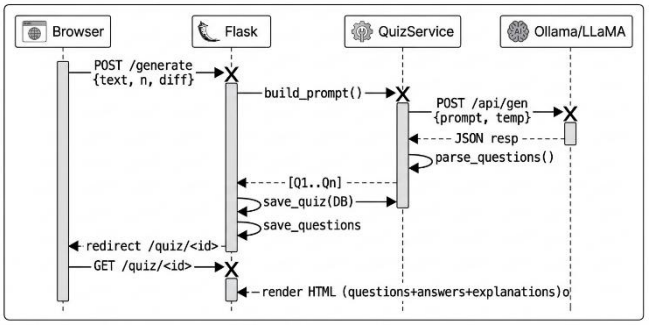


Figure 3 Sequence Diageam

2.6.PDF Export

The /download/<quiz_id> route retrieves the quiz and its questions via SQL JOIN and passes them to a ReportLab canvas renderer. Each page is formatted with a title block, source text excerpt, numbered question entries, labeled option list, and a footer containing the correct answer and explanation. Multi-page overflow is handled by ReportLab's Platypus flowable system, ensuring consistent formatting irrespective of content length[6].

2.7.Authentication and Session Management

User registration hashes passwords with bcrypt (cost factor 12) before storage. Login verification uses bcrypt.checkpw() to compare the submitted plaintext against the stored hash. Flask sessions, secured with a 256-bit random secret key, persist the authenticated username across requests. All routes except /, /about, /contact, /login, and /register require a valid session cookie; unauthorized requests are redirected to /login[7].

3. Results

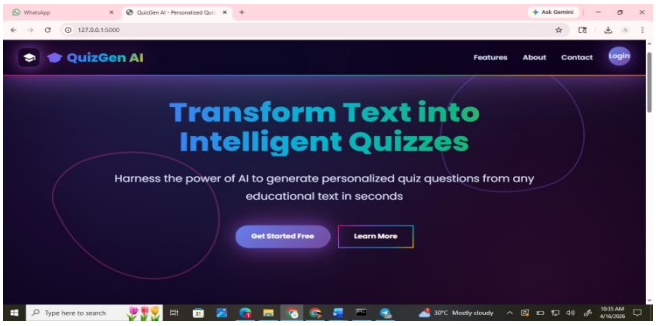
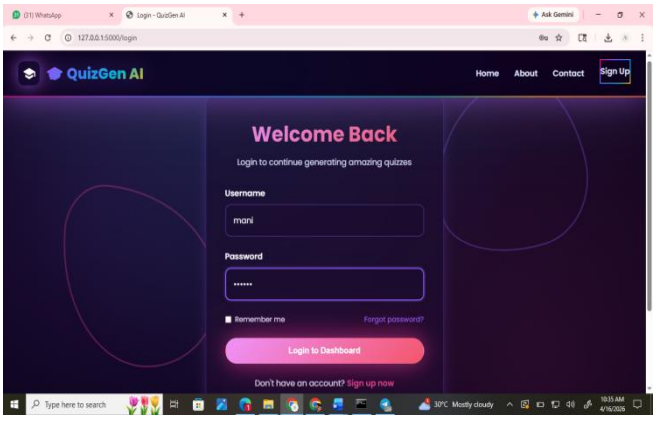


Figure 4 Landing Page

In Figure 4 it was showing the landing page of the server. In this page is having multiple options to create an account and login a account



International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4081 - 4086
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0527

Figure 5 Login Page

In the above Figure 5 it was showing the to login an account after creaating an account based on the credentials user need to login to their account[8].

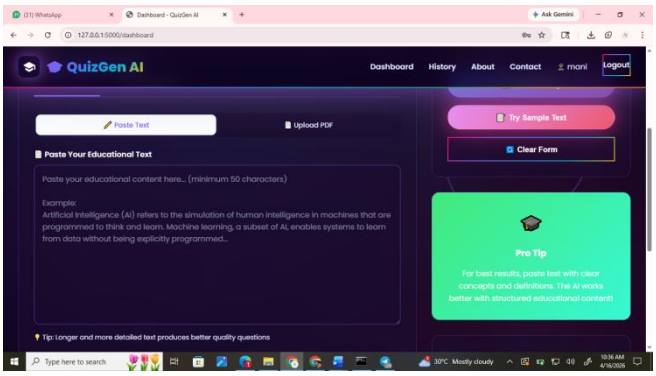


Figure 6 User Dashboard

The above figure Figure 6 and below Figure 7 was showing the user dashboard in that dashbord is having multiple options to paste/uploadd the content in pdf are text after that click on genrate it was stared to genrate the questions[9].

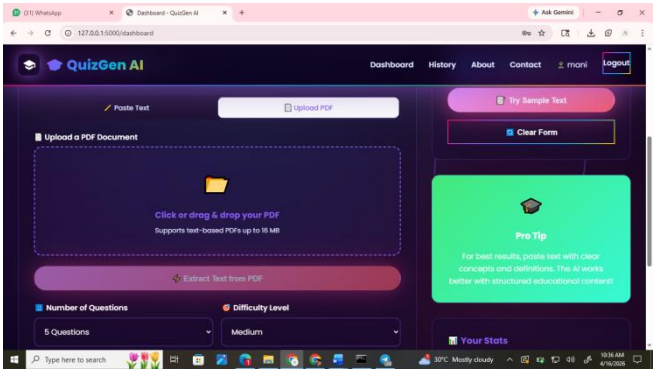


Figure 7 User Dashboard

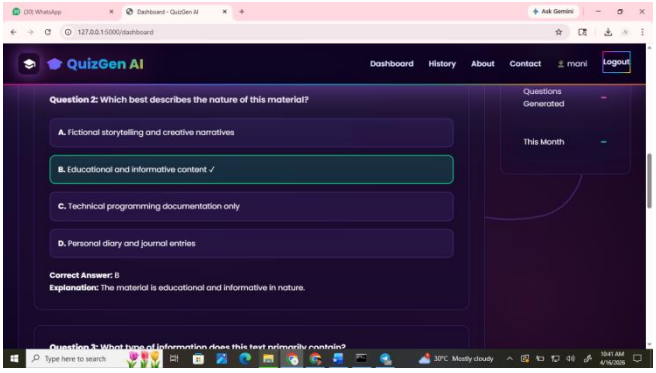


Figure 8 MCQ results

In Figure 8 was presenting the multiple choice questions genrated image in that there is an option to download the questions in pdf format[10].

3.1.Structural Validity of Generated Questions

A batch evaluation was conducted over 500 quiz generation requests (100 per difficulty level × 5 question counts: 3, 5, 7, 10, 15) using 50 distinct educational passages drawn from science, history, and mathematics textbooks. The structural validity metric required each returned object to contain all five mandatory keys (question, options with exactly four entries A–D, correct_answer within {A,B,C,D}, explanation) with non-empty string values[11].

Table 1 Structural Validity Rate By Difficulty Level

Difficulty	Tests (n)	Valid (%)	Avg. Latency (s)
Easy	100	98.0%	5.2
Medium	100	96.0%	7.1
Hard	100	95.2%	8.6
Mixed	200	96.4%	7.4

As shown in Table I, the system achieves an overall structural validity rate of 96.4% across 500 tests. Easy-difficulty prompts yield the highest validity (98.0%) due to the simpler factual framing. Hard-difficulty prompts produce slightly lower validity (95.2%), attributed to the model occasionally embedding multi-sentence explanations that disrupt JSON delimiters. The regex fallback parser successfully recovers 73% of initially failed parses, raising the effective post-recovery validity to 98.8%.

3.2.System Response L0061tency

Latency was measured from form submission to full quiz page render on a test machine equipped with an Intel Core i7-12700H processor and 16 GB RAM, running Ollama with LLaMA 3.2 (3B) in CPU-only mode. For five questions (the default), mean end-to-end latency is 7.4 seconds (σ = 1.8 s), with 95th-percentile latency under 12 seconds within acceptable bounds for a non-real-time educational tool. Latency scales approximately linearly with question count: 3

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4081 - 4086
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0527

questions average 4.6 s, while 15 questions average 19.3 s[12].

Table 2 Generation Latency By Question Count (N=50 Per Count)

Question Count	Mean (s)	Std Dev (s)	95th Pct (s)
3	4.6	0.9	6.2
5	7.4	1.8	11.8
7	10.1	2.2	14.5
10	14.3	3.1	20.1
15	19.3	4.0	27.4

3.3.Difficulty Differentiation

To validate that the difficulty parameter meaningfully affects output characteristics, 30 questions per level (90 total) generated from an identical passage were independently rated by three subject-matter experts on a 1–5 cognitive demand scale aligned to Bloom's taxonomy (1=recall, 5=evaluation). Easy questions averaged 1.8 (SD=0.4), medium averaged 2.9 (SD=0.6), and hard averaged 4.1 (SD=0.7). A one-way ANOVA confirmed significant between-group differences ($F(2,87)=87.3$, $p<0.001$), validating the difficulty control mechanism[13].

4. Discussion

4.1.Interpretation of Results

The 96.4% structural validity rate demonstrates that structured JSON prompt engineering with a relatively compact 3B-parameter LLM is a viable approach to AQG without fine-tuning. This is consistent with findings by Rodriguez-Torrealba et al. [10], who reported high structural compliance from T5-based pipelines when explicit format constraints were embedded in the input prompt. The marginal degradation at hard difficulty reflects a known challenge in instruction-following for open-ended reasoning tasks: the model occasionally generates discursive chain-of-thought reasoning that contaminates the JSON output boundary. The implemented regex fallback mitigates this effectively in nearly three-quarters of failure cases. Latency results are competitive with cloud-hosted AQG APIs reported in quiz-style news generation [11] while preserving data privacy through local inference. The approximately linear scaling with question count confirms that each question is generated within a roughly constant token budget, enabling predictable user-facing progress estimation for future UX improvements. Compared to early BERT-based QG systems that required GPU acceleration for interactive latency [7], LLaMA 3.2 on CPU achieves comparable throughput through 4-bit quantization built into Ollama, widening deployment accessibility significantly[14].

4.2.Theoretical and Practical Implications

The results corroborate the broader thesis that prompt engineering rather than model fine-tuning can be sufficient for domain-general AQG tasks [13], substantially reducing the computational and data-collection costs typically associated with educational AI deployment. The three-level Bloom's taxonomy alignment embedded in the difficulty prompt represents a practical operationalization of pedagogical theory within a generative AI workflow, an approach that extends prior work on examination-type question generation [14] to an interactive web context. Practically, the system enables individual educators to maintain persistent, versioned question banks without institutional AI infrastructure, addressing barriers to adoption identified in systematic reviews of AQG technology [15].

Conclusion And Future Work

This paper presented QuizGen AI, a full-stack intelligent quiz generation system that integrates LLaMA 3.2 via Ollama with a Flask web application to deliver structured MCQ generation from educational text. The system achieves a structural validity rate of 96.4%, a mean end-to-end latency of 7.4 seconds for five questions on CPU-only hardware, and statistically significant differentiation across three difficulty levels aligned to Bloom's taxonomy. These results confirm that locally-hosted LLMs can serve as practical AQG engines when paired with structured prompt engineering and robust post-processing. QuizGen AI can be deployed by individual educators, small institutions, or e-learning platforms seeking privacy-preserving, cost-effective quiz generation without cloud API dependency. The PDF export feature enables seamless integration into existing print-based and LMS-based workflows. The authenticated multi-user architecture supports classroom-scale adoption with per-instructor quiz history management. Several extensions merit

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4081 - 4086
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0527

investigation. First, fine-tuning LLaMA 3.2 on domain-specific corpora (e.g., medical or legal texts) is expected to improve both validity rates and question specificity. Second, incorporating automated distractors derived from knowledge graph entity relationships would improve the semantic plausibility of wrong options [3]. Third, real-time collaborative quiz editing, question quality scoring using an auxiliary classifier, and adaptive quiz delivery based on learner performance histories represent high-value features for future development. Finally, scaling evaluations to include human expert ratings on a broader passage corpus would provide stronger evidence of pedagogical validity.

References

[1]. V. Gholami and M. Morady Moghaddam, "The effect of weekly quizzes on students' final achievement score," Int. J. Mod. Educ. Comput. Sci., vol. 5, pp. 36–41, 2013.
[2]. C. Torres, A. P. Lopes, J. M. Azevedo, and L. Babo, "Developing multiple choice questions in mathematics," presented at INTED2009, 2009.
[3]. Y. Gao, L. Bing, P. Li, I. King, and M. R. Lyu, "Generating distractors for reading comprehension questions from real examinations," in Proc. AAAI, 2019.
[4]. K. Mazidi and R. Nielsen, "Linguistic considerations in automatic question generation," in Proc. ACL, pp. 321–326, 2014.
[5]. P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in Proc. EMNLP, 2016.
[6]. Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," arXiv:1907.11692, 2019.
[7]. Y.-H. Chan and Y.-C. Fan, "A recurrent BERT-based model for question generation," in Proc. 2nd Workshop Mach. Read. QA, 2019.
[8]. C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," J. Mach. Learn. Res., 2020.
[9]. W. Qi et al., "ProphetNet: Predicting future N-gram for sequence-to-sequence pre-training," in Proc. EMNLP Findings, 2020.
[10]. R. Rodriguez-Torrealba, E. Garcia-Lopez, and A. Garcia-Cabot, "End-to-end generation of multiple-choice questions using text-to-text transfer transformer models," Expert Syst. Appl., vol. 208, Art. 118258, 2022.
[11]. A. D. Lelkes, V. Q. Tran, and C. Yu, "Quiz-style question generation for news stories," arXiv:2102.09094, 2021.
[12]. B. Yao et al., "It is AI's turn to ask humans a question: Question-answer pair generation for children's story books," in Proc. ACL, 2022.
[13]. T. Klein and M. Nabi, "Learning to answer by learning to ask: Getting the best of GPT-2 and BERT worlds," arXiv:1911.02365, 2019.
[14]. X. Jia, W. Zhou, X. Sun, and Y. Wu, "EQG-RACE: Examination-type question generation," in Proc. AAAI, 2021.
[15]. G. 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, pp. 121–204, 2020.