International Journal of Interdisciplinary Innovative Research & Development (IJIIRD)
ISSN: 2456-236X
Vol. **11** Special Issue **03** | **2026**

Ai Quiz Generator — An Offline AI-Powered Quiz Generation and Intelligent Chat System Using llama.cpp

Tanmay B. Mestry[1], Sanjana K. Walavalkar[2], Shejal S. Gawade[3], Ranjeet B. Jadhav[4], Prof. Poonam Kadam[5]

[1,2,3,4] *Student, Computer Engineering Department, Metropolitan Institute of Technology and Management, Sindhudurg, Maharashtra, India*
[5]*Assistant Professor, Computer Engineering Department, Metropolitan Institute of Technology and Management, Sindhudurg, Maharashtra, India*

DOI: 10.5281/zenodo.20630195

ABSTRACT

In recent times, the rapid growth of e-learning and digital education has increased the demand for automated assessment tools. Educators and students require efficient tools to generate quizzes and assessments from their study material without depending on internet-based paid services. Existing cloud-based AI tools raise concerns related to data privacy, recurring API costs, and internet dependency, making them unsuitable for offline and resource-constrained educational environments.
Ai Quiz Generator is a full-stack offline AI-powered web application that enables users to automatically generate multiple-choice questions (MCQs) from their own text documents and images using a locally hosted Large Language Model (LLM) powered by llama.cpp. The system performs six key operations to ensure quality output: Generating structured MCQ questions from plain-text documents (.txt, .md), Generating questions from image and diagram inputs using vision models, Real-time AI chat with multi-turn conversation memory and session management, Seven-layer JSON repair to handle malformed model output, Persistent quiz history with score tracking and performance analytics, Secure user authentication using JWT and bcrypt password hashing

1. INTRODUCTION

In recent times, online education and digital learning platforms have grown significantly, driven by increased internet penetration, the COVID-19 pandemic's push toward remote learning, and the widespread availability of digital study material. In these digital learning environments, assessments and quizzes play an integral part in measuring student understanding and reinforcing knowledge retention. Creating high-quality multiple-choice questions (MCQs) manually requires considerable time and expertise from educators, making automated quiz generation a highly desirable capability.

Existing AI-powered quiz generation tools predominantly rely on cloud-based Large Language Model (LLM) APIs such as OpenAI's GPT-4. While these tools produce good results, they raise critical concerns. First, all uploaded study material — including private lecture notes, proprietary course content, and examination material — is processed on third-party servers, posing significant data privacy risks. Second, cloud API usage incurs recurring per-token costs that are prohibitive for individual students or small institutions. Third, these tools require a stable internet connection, making them inaccessible in bandwidth-limited or offline environments.

Thus, a locally hosted, privacy-preserving, AI-powered quiz generation system is needed that can generate MCQ questions from a user's own study material entirely offline. Ai Quiz Generator addresses this need by leveraging llama.cpp, an open-source high-performance C++ inference engine for quantized GGUF-format language models, enabling capable AI inference on consumer-grade hardware without any cloud dependency.

Many students and educators today create and share study material in various formats — text documents, scanned notes, slide screenshots, and diagram images — which is why Ai Quiz Generator supports both text and image inputs for quiz generation. On one hand, this makes learning more accessible; on the other, the challenge of extracting structured, semantically correct MCQ questions from raw LLM output must be addressed, as small language models frequently produce malformed JSON responses. That is why a robust seven-layer JSON repair and validation pipeline is a core contribution of this system.

Tasks of this project are performed in steps:

1. User logs in securely using email and password with JWT-based authentication.
2. User uploads a text document or image as quiz source material.
3. The system extracts content and constructs a structured prompt with a concrete JSON example.

4. The local llama.cpp LLM generates MCQ questions; the seven-layer repair pipeline validates the output.
5. Valid questions are stored in SQLite; user takes the quiz and receives a score.
6. Users can also interact with the AI through a real-time streaming chat interface for doubt resolution.
   Finally, the result is a complete, privacy-preserving, offline-capable learning platform.

2. PROBLEM STATEMENT

In the e-learning landscape, automated quiz generation significantly aids student assessment and self-study, but the dominance of cloud-based AI tools — which require internet access, charge per-token API fees, and process private educational data on third-party servers — threatens the privacy and accessibility of AI-assisted education. Existing tools cannot operate offline, support image-based inputs, or provide an integrated conversational AI interface alongside quiz generation. Manual quiz creation is time-consuming and impractical at scale, making it essential to deploy an offline, privacy-preserving Automated Quiz Generation System that leverages a locally hosted Large Language Model to process text and image inputs, generate structured MCQ questions with robust output validation, and provide a real-time conversational learning assistant — all without any cloud dependency, API cost, or internet requirement.

3. OBJECTIVES

- To develop a fully offline AI-powered system for generating MCQ quizzes from text and image inputs using a locally hosted GGUF language model via llama.cpp.
- To design and implement a seven-layer JSON repair algorithm achieving over 90% valid question extraction from raw LLM output, handling all known failure modes including markdown wrapping, truncation, and answer mismatches.
- To build a real-time streaming conversational AI chat interface with persistent multi-turn session history for interactive doubt resolution.
- To implement a secure, stateless authentication system using JWT tokens and bcrypt password hashing, protecting all user-specific data endpoints.
- To persist quiz history, scores, and chat sessions in a local SQLite database, enabling longitudinal performance analytics without any cloud storage.
- To design a model-agnostic architecture allowing any compatible GGUF model to be substituted through a single configuration file change, with no code modifications required.
- To deliver a production-quality multi-page frontend with zero build-step dependency, serving all pages from the same Express backend.

4. SYSTEM ARCHITECTURE

4.1 Technologies Used:

- Frontend: HTML5, CSS3, Vanilla JavaScript (ES2022), CSS Custom Properties
- Backend: Node.js v22, Express.js (REST API, MVC pattern)
- Database: SQLite 3 via better-sqlite3 (local, zero-configuration)
- AI Inference: llama.cpp (GGUF, CUDA/CPU), Qwen2.5-3B-Instruct-Q4_K_M (default model)
- Authentication: JSON Web Tokens (JWT, HS256, 30-day expiry), bcryptjs (cost factor 12)
- File Upload: Multer (multipart/form-data, disk storage)
- Streaming: Server-Sent Events (SSE) via Axios readable stream + Express res.write()
- Fonts: DM Sans, DM Mono, Bebas Neue (Google Fonts)

4.2 Modules:

- User Authentication Module — Registration, login, JWT issuance, and protected route middleware
- Quiz Generation Module — File upload, prompt engineering, LLM invocation, JSON repair, question validation, and score recording
- Chat Module — Session management (CRUD), multi-turn message history, SSE token streaming, and image attachment support
- User Profile Module — Profile editing, avatar customization, password change, and account deletion with cascade cleanup
- Analytics and History Module — Aggregated statistics (average score, total quizzes, best performance, model usage breakdown) and paginated quiz history

4.3 Three-Tier Architecture Overview:

Presentation Layer: Browser-based vanilla JS multi-page frontend. Seven pages: Login, Register, Dashboard, Generate, History, Chat, Profile. Central api/api.js module handles all API calls with JWT injection.

Application Layer: Node.js Express backend (port 3001). MVC structure: Routes define endpoints; Middleware handles JWT verification and file upload; Controllers implement business logic; Models encapsulate all SQL operations.

Data and Inference Layer: SQLite database (quizmind.db) for all persistent data. llama.cpp server (port 8080) provides OpenAI-compatible /v1/chat/completions endpoint for LLM inference.

4.4 Methodology

- The user registers with a valid email and password (minimum 6 characters). The password is hashed using bcryptjs at cost factor 12 before storage in SQLite. On login, credentials are verified against the hash and a JWT token (30-day validity) is issued and stored in the browser's localStorage for subsequent authenticated API calls.
- The user uploads a text document (.txt, .md) or an image file (PNG, JPG) on the Generate page. Multer middleware validates the file MIME type and size (5 MB limit for text, 10 MB for images) and saves it to a temporary uploads directory.
- The quizController reads the uploaded file. For text files, the first 800 characters are extracted as an excerpt. For image files, the binary is base64-encoded. A structured prompt is built with a one-shot JSON example to guide the small language model toward structured output. The prompt is sent to the llama.cpp server via Axios streaming.
- The raw LLM output is passed through the seven-layer JSON repair pipeline: markdown fence stripping, bracket isolation, direct JSON.parse, truncated array recovery, regex object extraction, block scanning, and fallback. Valid question objects are further validated — options are padded to four, answer field mismatches are auto-corrected, and questions with missing fields are discarded.
- Valid MCQ questions are saved to the quiz_history table with metadata (model name, difficulty, total questions). The user answers the quiz interactively in the browser; upon submission, the score and time taken are saved via POST /api/quiz/score. Analytics are recalculated on the Dashboard.
- For the Chat feature, the user selects or creates a session. Each message is appended to the session history and the full conversation array is sent to llama.cpp as the messages parameter. The streamed response is proxied to the browser via SSE, token by token. The assistant response is saved to the chat_messages table on completion. Sessions persist across browser refreshes, and the sidebar lists all sessions ordered by last activity.

5. RESULTS & DISCUSSION

The implementation of Ai Quiz Generator yielded highly promising results in delivering a fully offline, privacy-preserving, AI-powered educational platform on consumer hardware. By integrating a locally hosted quantized Large Language Model with a robust seven-layer JSON repair pipeline and a streaming conversational chat interface, the system successfully demonstrated practical viability as a zero-cost alternative to cloud-based quiz generation tools.

On a GPU-equipped machine (NVIDIA RTX 3060 6 GB VRAM), average quiz generation latency was 18.4 seconds for 3 questions and 28.6 seconds for 5 questions — well within the 60-second target. The seven-layer JSON repair algorithm achieved a 92.4% overall parse success rate across 800 test runs, significantly exceeding the 85% target. Layer 3 (direct JSON.parse after preprocessing) resolved 71.4% of all cases, confirming that markdown fence wrapping and preamble text — the two most common failure modes — are effectively eliminated by preprocessing alone.

Discussions revealed that the one-shot prompting strategy — including a complete JSON example in the user prompt — improved raw model compliance from 51% (zero-shot) to 73% before repair, a 43% relative improvement. The model-agnostic architecture was validated by successfully swapping from Qwen2.5-3B to Phi-3.5-mini and MiniCPM-V-2.6 (vision) with only three .env file changes. The system met all eight stated objectives and demonstrated strong potential for adoption in offline educational settings.

Performance Summary Table:

Metric	Target	Achieved (GPU)	Achieved (CPU)
Quiz Generation (5 Qs)	≤ 60 seconds	28.6 seconds	134.8 seconds
JSON Repair Success Rate	≥ 85%	92.4%	92.1%
Chat First-Token Latency	≤ 3000 ms	1,840 ms	8,200 ms
Chat Token Throughput	≥ 20 tok/s	38.6 tok/s	8.4 tok/s
Database Query Time	≤ 10 ms	0.4 ms avg	0.4 ms avg
Authentication Latency	≤ 300 ms	187 ms	185 ms
Backend Memory (RSS)	≤ 150 MB	67 MB	65 MB
File Rejection Accuracy	100%	100%	100%

6. ADVANTAGES OF PROPOSED SYSTEM

- Users generate high-quality MCQ quizzes from their own private study material entirely offline — no internet connection, no API key, and no recurring cost required.
- All uploaded documents and images are processed locally on the user's machine; no data is transmitted to external servers, ensuring complete data privacy and GDPR/FERPA compliance.
- The seven-layer JSON repair algorithm ensures robust quiz generation even when the small language model produces malformed output, making the system reliable for everyday academic use.
- Users can interact with the integrated AI chat assistant for real-time doubt resolution, question explanations, and Socratic dialogue — all backed by the same local language model.
- Complete quiz history, scores, and performance analytics are persisted locally in SQLite, allowing students to track their learning progress over time without any cloud account.
- The model-agnostic architecture allows educators and advanced users to swap to any GGUF-compatible model — from lightweight 1B-parameter models for low-RAM devices to powerful 14B-parameter models for high-quality output — by editing a single configuration file.
- Image and diagram inputs are supported via vision models (MiniCPM-V, LLaVA), enabling quiz generation from scanned notes, textbook photographs, and slide screenshots — a capability not found in any existing offline tool.

7. CONCLUSION

Generating high-quality structured assessments from unstructured educational content using locally hosted AI models has emerged as a significant research and engineering challenge, and this project addresses it comprehensively. Although various LLM-based quiz generation approaches have been explored in research, most depend on cloud APIs that compromise privacy and incur costs; no existing system integrates offline inference, multi-modal input, conversational AI, and robust output repair in a single deployable application.

Ai Quiz Generator successfully demonstrates that a fully offline, privacy-preserving, AI-powered educational platform is practically achievable on consumer hardware. The system will assist students and educators in generating reliable MCQ assessments from their own private study material without falling into the trap of cloud dependency, API costs, or data privacy risks. Users can acquire quizzes from both text documents and images, receive real-time AI assistance through the integrated chat interface, and track their performance over time through persistent local analytics — all with a single local installation requiring no internet access after setup.

The seven-layer JSON repair algorithm achieves 92.4% valid question extraction — a key contribution that makes small, quantized language models practically usable for structured output generation tasks where prior implementations failed. The model-agnostic architecture with single-file configuration ensures the system remains relevant as new and more capable GGUF models are released. Our project's goal of improving learning efficiency while making AI-assisted education private, accessible, and cost-free is demonstrated through quantitative benchmarks across multiple hardware configurations.

It is feasible to further improve question quality by integrating larger GGUF models as hardware capabilities grow, by implementing PDF support for broader document compatibility, and by adding Bloom's Taxonomy filtering for curriculum-aligned assessment generation. The system is positioned as a strong foundation for a next-generation offline intelligent tutoring system that respects user privacy while delivering commercial-grade AI capabilities.

REFERENCES

[1] Heilman, M., & Smith, N. A. (2010). Good Question! Statistical Ranking for Question Generation. Proceedings of NAACL-HLT 2010.

[2] Du, X., Shao, J., & Cardie, C. (2017). Learning to Ask: Neural Question Generation for Reading Comprehension. Proceedings of ACL 2017.

[3] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

[4] Brown, T. B., et al. (2020). Language Models are Few-Shot Learners (GPT-3). Advances in Neural Information Processing Systems (NeurIPS 2020).

[5] Raffel, C., et al. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research, 21(140).

[6] Ouyang, L., et al. (2022). Training Language Models to Follow Instructions with Human Feedback (InstructGPT). NeurIPS 2022.

[7] Touvron, H., et al. (2023). LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

[8] Gerganov, G. (2023). llama.cpp: High-Performance Inference of LLaMA models in C/C++. GitHub Repository. https://github.com/ggerganov/llama.cpp

[9] Frantar, E., Ashkboos, S., Hoefler, T., & Alistarh, D. (2022). GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. arXiv:2210.17323.

[10] Qwen Team, Alibaba Cloud. (2024). Qwen2.5 Technical Report. arXiv:2412.15115.

[11] Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. EMNLP 2016.

[12] Zhao, Y., et al. (2018). Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks. EMNLP 2018.