AssessMate: A Retrieval-Augmented Framework for Syllabus-Grounded Assessment Generation And AI-Assisted Learning

Pavan S. Kumar[1], Ragavan B.[2], Rampraveen S.[3], Sangeeth R.[4], Sai Sriram B. M.[5], P. Deepa[6]
[1,2,3,4,5,6]*Department of Computer Science and Engineering, SNS College of Technology, Coimbatore, Tamil Nadu, India*

***Abstract*—Background: Secondary-school assessment often separates textbook study, test preparation, answer evaluation, and progress tracking. Objective: This paper presents AssessMate, an AI-integrated learning and assessment platform for CBSE and State Syllabus students in Classes X and XII. Methods: The platform combines retrieval-augmented generation (RAG), configurable tests, AI-assisted answer evaluation, performance analytics, a digital textbook library, and the Mate AI study assistant. Textbook PDFs are divided into approximately 500-token chunks with 50-token overlap, tagged with curriculum metadata, embedded using BAAI/bge-small-en-v1.5, and indexed in ChromaDB. Retrieved passages provide context to Google Gemini for question generation and learning assistance. Results: Documented functional checks cover authentication, test configuration, question generation, submission, evaluation, analytics, and learning support. Conclusion: AssessMate integrates syllabus-aware assessment and learning workflows. Quantitative validation of retrieval quality, grading reliability, usability, and learning outcomes remains future work.**

***Index Terms*—artificial intelligence in education, automated assessment, educational technology, learning analytics, retrieval-augmented generation**

I. INTRODUCTION

Assessment is an important part of secondary education because it helps learners identify gaps in understanding and helps teachers plan follow-up instruction. For Class X and Class XII students, effective preparation requires more than a final score: students benefit from explanations of errors, practice aligned with the prescribed curriculum, and visibility into performance across topics. Teachers, meanwhile, need practical ways to prepare assessments, manage submissions, and review results without repeating time-consuming manual work.

Many learning workflows distribute textbooks, question banks, online tests, answer checking, and progress records across different tools. General-purpose generative language models can create questions and explanations quickly, but their outputs may contain unsupported information or drift away from a particular syllabus. Subjective answer evaluation introduces another challenge because responses may express correct concepts in different words, include multi-step reasoning, or be submitted as handwritten images.

AssessMate addresses these issues by integrating textbook retrieval with generative AI and assessment management in one platform. The design follows a diagnostic feedback loop: select learning context, retrieve relevant textbook content, generate an assessment, attempt questions, evaluate responses, inspect performance, and use feedback to guide further learning. The platform is intended to assist students and teachers, not replace teacher judgement.

The main contributions of this paper are: (1) a curriculum-aware RAG workflow for generating assessments from indexed textbook resources; (2) an integrated evaluation pipeline for objective, typed subjective, and supported handwritten responses; (3) student and teacher workflows connected to assessment history and diagnostic analytics; and (4) a practical full-stack implementation with structured outputs and model fallback handling.

A. Related work

Information retrieval provides established methods for indexing collections and finding text relevant to a user query [1]. In an educational setting, retrieval can

constrain generated questions and explanations to prescribed learning materials. Retrieval-augmented generation combines a retriever with a generative model, allowing the model to condition its output on retrieved passages rather than relying only on parameters learned during pretraining [2]. This pattern is relevant to curriculum-bound tasks, although retrieval alone does not guarantee that every generated statement is correct.

Research on intelligent tutoring systems emphasizes interactive support, feedback, and student-centered learning rather than one-way content delivery [3]. Learning analytics studies how learner data can be collected and interpreted to understand learning processes and inform educational decisions [4]. These ideas motivate the integration of contextual assistance and progress summaries into AssessMate. Broader reviews of artificial intelligence in education similarly note that educational context and educator involvement matter when assessing these systems [5]. Automated assessment systems can efficiently score objective responses, but open-ended answers require interpretation of concepts, completeness, and reasoning.

Language-model-based scoring can support feedback generation, yet consistency, bias, explainability, and agreement with human evaluators must be tested before high-stakes use. Image-based handwritten answer evaluation adds uncertainty from image quality and text recognition. Accordingly, AssessMate treats subjective and handwritten grading as AI-assisted and identifies teacher review and quantitative validation as important safeguards.

The distinctive focus of AssessMate is the integration of these capabilities around selected board, class, subject, and chapter metadata, rather than presenting a standalone question generator or chatbot.

B. System objectives and scope

AssessMate is designed for CBSE and State Syllabus resources for Classes X and XII. Its objectives are to reduce fragmentation between learning resources and assessments, provide configurable practice, support teacher-created tests, deliver timely feedback, and organize performance history for subsequent review.

- Assessment generation: configurable board, class, subject, chapter, subtopic, difficulty, question type, and test mode.
- Question formats: multiple-choice, short-answer, case-based, and long-answer questions.
- Evaluation: answer-key comparison for MCQs and rubric-guided AI evaluation for subjective responses, including supported handwritten submissions.
- Analytics and reporting: scores, accuracy, efficiency-related indicators, critical-thinking scores, assessment history, and detailed reports.
- Teacher controls: test creation, access codes, expiration dates, reattempt settings, submission monitoring, and review of evaluation results.
- Mate AI: contextual hints and concept clarification intended to support student reasoning rather than directly reveal answers.

II. METHODS

A. System architecture

The platform uses a modular client-server architecture. The Next.js 14 frontend, implemented with TypeScript and Tailwind CSS, provides the student and teacher interfaces. The FastAPI backend exposes REST endpoints for authentication, assessment generation, evaluation, textbook access, and Mate AI. SQLite stores account information, test history, teacher-created tests, and chat history through SQLAlchemy.

The retrieval component accesses textbook vectors stored in ChromaDB Cloud or a local persistent store, while Google Gemini API models provide question generation, subjective evaluation, vision-assisted handwritten response processing, and contextual chat. Provider documentation should be consulted to confirm current model identifiers, API behavior, and service limits [7]. Chroma's documentation describes the vector-store capabilities used for retrieval [8].

The separation of interface, API, database, retrieval, and model services allows the components to be maintained independently. Requests are validated using structured schemas, and generated assessments are expected to conform to JSON structures that the frontend can render. A model fallback chain is configured to handle selected service errors and rate limits. These mechanisms improve operational resilience but do not, by themselves, establish semantic correctness or educational validity.

B. Retrieval-augmented question generation

The RAG workflow begins when a learner selects a curriculum board, class, subject, chapter, and assessment configuration. The backend constructs a retrieval query and applies curriculum metadata filters so that the selected textbook collection is used. Textbook PDF content is divided into chunks of approximately 500 tokens with an overlap of approximately 50 tokens. Each chunk is associated with metadata including board, class level, subject, and chapter. The BAAI/bge-small-en-v1.5 model converts chunks and search queries into embedding vectors, which are indexed in ChromaDB. The vector-store configuration follows the capabilities described in the Chroma documentation [8]. Sentence-embedding methods provide a general foundation for semantic retrieval, although the specific embedding model used here is BAAI/bge-small-en-v1.5 [6], [10]. At generation time, the retriever returns relevant chunks for the selected academic context. These passages are included in a structured prompt to a Gemini model, which generates an assessment in the required format. Prompt instructions encourage the model to stay within retrieved textbook context, and schema validation is used to reduce malformed output. If a model endpoint is unavailable or rate-limited, the configured fallback sequence can attempt another supported model.

The workflow is summarized as: textbook PDFs → chunking and metadata assignment → embedding and vector indexing → query embedding and filtered retrieval → context-grounded prompt → structured assessment output. Retrieval quality depends on the completeness of source textbooks, chunk boundaries, metadata accuracy, and the relevance of the returned passages. Future evaluation should measure retrieval precision and recall, the proportion of questions supported by retrieved evidence, and the rate of unsupported claims.

The documented implementation initially used a local sentence-transformers model. Deployment constraints on a low-memory hosting tier motivated a move toward cloud-based embedding calls to reduce the runtime memory footprint. This illustrates a practical trade-off between local model control, service dependence, latency, cost, and deployment resource limits.

C. Answer evaluation and learning support

AssessMate separates objective and subjective evaluation. Multiple-choice answers are compared with the stored answer key, enabling deterministic scoring for that question type. Subjective answers are evaluated by a Gemini model using the question, expected answer or marking scheme, relevant context, and an explicit rubric. The documented rubric assigns 60% to conceptual accuracy, 20% to completeness and key points, and 20% to clarity and terminology. The resulting feedback can identify missing concepts, incorrect reasoning, omitted formula steps, and areas for improvement.

For supported handwritten submissions, the student uploads an image of an answer sheet. The multimodal model processes the image and evaluates the response in the context of the question and marking criteria. Image clarity, handwriting legibility, layout, and mathematical notation can affect interpretation. Therefore, image-based grading should be treated as assistive and should remain reviewable by a teacher, especially when marks have consequential effects.

Mate AI provides contextual hints and concept explanations within the study workflow. Its intended role is to help the learner reason through a problem rather than immediately disclose a complete answer. Its usefulness depends on accurate retrieval, appropriate prompting, and alignment between the conversation and the learner's selected academic context.

The platform records assessment outcomes so that students can inspect their history and view metrics such as score and accuracy. Additional indicators, including efficiency and critical-thinking scores, are product-defined metrics; their interpretation and educational validity require clear definitions and empirical study before comparisons across learners or assessments are treated as standardized measurements.

D. Implementation and deployment

The frontend is built with Next.js 14, React, TypeScript, and Tailwind CSS. It includes landing, authentication, dashboard, test builder, test attempt, result, and teacher portal views. KaTeX-based rendering supports mathematical notation. The backend uses Python 3.10 or later and FastAPI, with separate routers for authentication, tests, chat, textbooks, and administrative statistics. Pydantic schemas validate API inputs and structured responses.

SQLite and SQLAlchemy manage users, assessment history, teacher tests, and Mate AI chat history. JWT-based authentication supports protected routes and student/teacher roles. The documented implementation stores access tokens on the client and sends them in bearer authorization headers; deployment should use secure token handling, short-lived credentials where appropriate, and careful protection of personal and academic data. API keys and service credentials should be supplied through environment configuration rather than committed to source control.

The application is designed for separate frontend and backend deployment, with the frontend hosted on Vercel and the API on Render. A documented deployment issue involved memory consumption from loading PyTorch and sentence-transformers within a 512 MB service limit. The project report describes a cloud-embedding approach as an optimization. Exact production model identifiers and API availability should be verified at the time of deployment because provider model names and endpoints can change.

III. RESULTS

The project report describes unit and functional checks across authentication, assessment configuration, question generation, answer submission, evaluation, analytics, reports, textbook retrieval, and Mate AI. Example checks include valid and invalid login inputs, selection of assessment modes, generation for a selected class and chapter, storing submissions, updating result history, and returning contextual responses. The report also describes verification of mathematical rendering and the configured question allocations for supported test presets.

These checks provide implementation-level evidence that the main workflows were considered and exercised. However, the supplied documentation does not provide a reproducible test dataset, sample size, timing measurements, inter-rater agreement, question-grounding scores, or a controlled comparison with human grading. Consequently, this paper does not claim measured gains in learning outcomes, grading accuracy, teacher workload, or system performance. Such claims would require a separate empirical evaluation.

A suitable next evaluation would use a held-out collection of textbook sections and questions, assess retrieval relevance, and ask subject teachers to rate question syllabus alignment, correctness, difficulty, and clarity. For subjective answers, AI scores should be compared with independent teacher scores using agreement metrics and error analysis. Handwritten submissions should be stratified by image quality and answer type. Usability studies with consented student and teacher participants could examine task completion, perceived usefulness, and accessibility.

IV. DISCUSSION

Generative models can produce incorrect questions, unsupported explanations, or inconsistent partial-credit decisions even when prompts include retrieved context. Retrieval can return irrelevant or incomplete passages, and strict JSON output only addresses structural validity, not factual validity. Rubric-guided grading may also penalize valid alternative phrasing or reward superficially similar but conceptually incorrect answers.

The platform handles accounts, submissions, academic performance data, and potentially identifiable answer-sheet images. Data minimization, access control, secure transport, retention policies, and transparent disclosure of AI processing are important. Student work should not be used to make consequential decisions solely through automated scores without appropriate human oversight. Curriculum resources should be indexed and distributed in accordance with their applicable access and copyright terms.

The current report documents the system’s architecture and functional workflows but does not establish large-scale reliability or educational effectiveness. Adaptive testing, additional language support, expanded analytics, stronger teacher override controls, and broader resource coverage remain potential areas for development.

V. CONCLUSION AND FUTURE WORK

This paper presented AssessMate, an AI-integrated assessment and learning platform for Class X and Class XII students and teachers using CBSE and State Syllabus resources. Its design connects curriculum metadata and textbook retrieval with configurable test

generation, online assessment, objective and AI-assisted subjective evaluation, analytics, reporting, and contextual study support through Mate AI. The implementation combines a Next.js frontend, FastAPI backend, SQLite persistence, ChromaDB retrieval, BAAI embeddings, and Google Gemini services. The vector-store configuration follows the capabilities described in the Chroma documentation [8].

The principal contribution is an integrated workflow that attempts to keep assessment generation and learning assistance anchored to selected textbook context while connecting generated tests to evaluation and progress history. The documented functional checks cover major system modules, but the present evidence is not sufficient to claim quantified improvements in educational outcomes or grading reliability. Future work should establish benchmark datasets, measure retrieval and generation quality, compare AI grading with teacher judgement, assess handwritten-answer robustness, and conduct structured usability and learning-outcome studies. Teacher review and transparent limitations should remain central to the use of AI-assisted evaluation.

ACKNOWLEDGMENT

The authors acknowledge the Department of Computer Science and Engineering, SNS College of Technology, Coimbatore, Tamil Nadu, India, for providing the academic environment and institutional support for the development of AssessMate.

AUTHOR CONTRIBUTIONS

CONFLICT OF INTEREST

FUNDING

DATA AVAILABILITY

ETHICS STATEMENT

The manuscript does not report a human-participant study or a newly collected participant dataset. If user testing or participant data were collected, the manuscript must report the applicable institutional ethics approval or exemption and consent procedures. This paper describes a software implementation and does not report a newly collected research dataset. Textbook materials used by the platform remain subject to their source-specific access and reuse conditions. The authors should provide a repository link for code or evaluation artefacts if these are intended to be publicly available.

No external funding information was provided in the project materials. The authors should confirm whether the project received external funding and report the source and grant details, if applicable.

The authors should disclose any financial or non-financial competing interests relevant to this work. If none exist, the author group may state: “The authors declare no conflict of interest.”

Pavan S. Kumar was responsible for the software implementation and initial manuscript preparation. The remaining authors are listed as co-authors of this team project; their specific intellectual contributions must be accurately confirmed by the author group before submission. No individual contribution is attributed here without supporting confirmation.

REFERENCES

[1] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge University Press, 2008, doi: 10.1017/CBO9780511809071.

[2] P. Lewis *et al*., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[3] B. P. Woolf, *Building Intelligent Interactive Tutors: Student-Centered Strategies for Revolutionizing E-Learning*. Burlington, MA, USA: Morgan Kaufmann, 2009.

[4] G. Siemens and R. S. J. d. Baker, “Learning analytics and educational data mining: Towards communication and collaboration,” in *Proc. 2nd Int. Conf. Learning Analytics and Knowledge (LAK ’12)*, 2012, pp. 252–254, doi: 10.1145/2330601.2330661.

[5] O. Zawacki-Richter, V. I. Marín, M. Bond, and F. Gouverneur, “Systematic review of research on artificial intelligence applications in higher education—Where are the educators?” *Int. J. Educ. Technol. High. Educ.*, vol. 16, Art. no. 39, 2019, doi: 10.1186/s41239-019-0171-0.

[6] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” in *Proc. 2019 Conf. Empirical Methods in Natural Language Processing and 9th Int. Joint Conf. Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992, doi: 10.18653/v1/D19-1410.

[7] Google, “Gemini API documentation,” *Google AI for Developers*. [Online]. Available: Gemini API Documentation. Accessed: Oct. 26, 2026.
[8] Chroma, “Chroma documentation.” [Online]. Available: Chroma Documentation. Accessed: Oct. 26, 2026.
[9] National Council of Educational Research and Training, “Textbooks.” [Online]. Available: NCERT Textbooks. Accessed: Oct. 26, 2026.
[10] BAAI, “bge-small-en-v1.5,” *Hugging Face Model Repository*. [Online]. Available: BGE Small English v1.5 Model. Accessed: Oct. 26, 2026.