International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 22, No. 2(1), 2026

AI-ENABLED REVIEW MANAGEMENT SYSTEM

AN INTEGRATED PLATFORM FOR AI-ASSISTED ACADEMIC SUBMISSION REVIEW

K. Harshwardhan, M. Navya Vani, V. Yaswanth Varma, N. Veneela, K. Karthik
Department of Computer Science and Engineering
Raghu Engineering College
Visakhapatnam, India
harshwardhan.kapu@gmail.com

Dr. Uttam Mande
Professor, Department of Computer Science and Engineering
Raghu Engineering College
Visakhapatnam, India
uttammande@gmail.com

***Abstract*—The proliferation of digital academic and conference events demands scalable, consistent, and transparent submission review processes. This paper presents an AI-Enabled Review Management System (AI-RMS) built upon a Node.js/Express.js REST API backend with MongoDB persistence, integrating a Gemini-powered AI review engine to generate structured feedback for student submissions. The system supports three distinct roles—Administrator, Reviewer, and Student—governed by JWT-based authentication and role middleware. A quantitative reviewer rating model combining accuracy (60%), timeliness (30%), and workload (10%) enables rating-based workload distribution. Experimental deployment demonstrates reduced manual review latency, improved feedback consistency, and actionable reviewer performance analytics.**

Keywords*—Review Management; Artificial Intelligence; Node.js; Express.js; MongoDB; JWT Authentication; Gemini API; Role-Based Access Control

I. INTRODUCTION

Academic institutions and professional organizations conduct a significant number of events—seminars, hackathons, project exhibitions, and conferences—that involve the collection and evaluation of student submissions. Managing these processes manually introduces bottlenecks: inconsistent feedback quality, delayed reviewer responses, uneven workload distribution, and limited transparency for participants. An automated, AI-assisted review management platform addresses these pain points by streamlining submission intake, AI-based preliminary evaluation, and human reviewer oversight within a unified system.

The AI-Enabled Review Management System (AI-RMS) described in this paper provides a full-stack web application that orchestrates the lifecycle of an academic event from creation through final review decision. Administrators create events with configurable registration and submission windows, assign reviewers using a rating-aware distribution algorithm, and monitor aggregate submission status. Reviewers receive AI-generated preliminary feedback and decide to accept or reject it, or override with manual assessments. Students join events via unique event codes, upload submissions, and receive structured feedback.

The key contributions of this work are: (1) a modular REST API architecture built on Express.js 5.x with role-based access control enforced by JWT middleware; (2) integration of the Gemini multimodal AI model for automated submission review supporting PDF, image, text, and code file formats; (3) a quantitative reviewer rating model that drives rating-aware workload distribution; and (4) a date-driven event lifecycle state machine managing transitions between upcoming, registration, submissions, active, and completed states.

II. RELATED WORK

Peer review and assignment management systems have been studied extensively in educational technology research. EasyChair and HotCRP are widely deployed conference management platforms that provide submission, assignment, and review workflows but lack integrated AI feedback capabilities [1]. Coursera and edX incorporate automated grading for structured assignments using rubric-based scoring, yet do not generalize to open-ended, domain-agnostic submission types common in academic events [2].

Recent advances in large language models (LLMs) have motivated their application to automated assessment. Ramesh and Sanampudi demonstrated that GPT-based models can produce evaluations

International Journal of Engineering Research and Science & Technology

correlating with human graders on short-answer questions, though domain-specific calibration is required [3]. Multimodal LLMs such as Google Gemini extend this capability to PDF and image inputs, enabling evaluation of design submissions and poster presentations that text-only models cannot process [4].

Reviewer assignment algorithms range from simple round-robin distribution to preference-based matching using bipartite graph algorithms [5]. Rating-aware assignment—where higher-rated reviewers receive additional workload proportional to demonstrated quality—has been proposed as a fairness-efficiency tradeoff in peer review literature [6]. The AI-RMS implements a pragmatic variant of this approach, sorting reviewers by computed rating and assigning remainder submissions to the top-rated reviewer, providing a deployable approximation within a standard web application framework.

III. SYSTEM OVERVIEW & DATA MODEL

A. Technology Stack

The AI-RMS is implemented as a single-server Node.js application using Express.js 5.2.1 as the HTTP framework. Data persistence is handled by MongoDB accessed via Mongoose 9.2.4 ODM. Authentication tokens are issued as JSON Web Tokens (JWT) verified on each protected route by a dedicated authMiddleware module. File uploads are processed using the Multer middleware, storing files on the local filesystem with paths recorded in the Submission document.

TABLE I. System Technology Stack

Component	Technology	Version
Backend Runtime	Node.js	LTS
HTTP Framework	Express.js	5.2.1
Database ODM	Mongoose	9.2.4
AI Engine	Gemini API	Multimodal
Auth Mechanism	JWT	HS256
File Handling	Multer	Latest

B. Data Models

Three Mongoose schemas define the data layer. The User schema stores identity, hashed credentials, role enumeration (admin, reviewer, student), and reviewer-specific rating fields (rating, totalReviews, avgAccuracy, avgTimeliness, technicalDomains). The Event schema captures event metadata, an eventCode unique identifier, phase date windows (registrationStartDate, registrationEndDate, submissionStartDate, submissionEndDate), and embedded arrays for enrolled students, assigned reviewers, and student-reviewer assignment pairs. The Submission schema links each submission to its parent event and submitting student, records filePath and fileName, stores aiFeedback and reviewerFeedback text, and tracks status through an enumerated lifecycle: pending, ai-reviewed, AI Review Accepted, AI Review Rejected.

TABLE II. Event Lifecycle Status States

Status	Description
upcoming	Event created, registration not yet open
registration	Registration window active
submissions	Submission window active
active	Ongoing event without defined submission end
completed	Submission window has closed

IV. METHODOLOGY

A. Role-Based Access Control

Access control is implemented through two middleware layers. The authMiddleware module verifies the Bearer JWT token present in the Authorization header, decodes the payload, and attaches the user object to req.user. The roleMiddleware module accepts a list of permitted roles and returns a 403 Forbidden response if req.user.role is not in the allowed set. All routes are protected by at least authMiddleware; administrative and reviewer-specific routes additionally require roleMiddleware(["admin"]) or roleMiddleware(["reviewer"]) respectively. Students access submission and event-join routes via roleMiddleware(["student"]).

B. AI Review Engine

When a reviewer triggers the AI review action, the submissionController.reviewSubmission handler reads the submission file from the filesystem. File type branching selects the appropriate Gemini input format: PDF and image files are base64-encoded and supplied as inlineData content parts enabling Gemini's multimodal parsing; text-based files (txt, md, html, js, css, csv) are read as UTF-8 strings and truncated to 25,000 characters to remain within token limits. The event title and description are passed as context so the AI can evaluate relevance to the event objectives. The Gemini model returns structured feedback, which is stored in aiFeedback and the submission status transitions to ai-reviewed.

C. Reviewer Rating Model

The reviewer rating algorithm in reviewerService.js computes a composite rating after each review decision. Accuracy is defined as max(0, 10 − |reviewerScore − aiScore|), penalizing reviewers whose manual scores deviate significantly from the AI baseline. Timeliness begins at 10 and decreases by 1 point for each 24-hour period of delay beyond the 48-hour review window. Running averages for accuracy and timeliness are updated incrementally

International Journal of Engineering Research and Science & Technology

using the recurrence newAvg = (oldAvg × n + newValue) / (n+1). The final rating formula is:

Rating = 0.6 × avgAccuracy + 0.3 × avgTimeliness + 0.1 × totalReviews

TABLE III. Reviewer Rating Formula Parameters

Parameter	Weight	Max Value	Description
avgAccuracy	0.60	10	Agreement with AI score
avgTimeliness	0.30	10	Speed of review completion
totalReviews	0.10	Unbounded	Reviewer experience proxy

D. Rating-Aware Reviewer Assignment

When an administrator assigns reviewers to an event, the assignReviewer controller fetches all reviewer User documents sorted by rating in descending order. Students are distributed floor(N/R) per reviewer, where N is the number of enrolled students and R is the number of reviewers. The remainder N mod R additional students are assigned to the highest-rated reviewer, incentivizing review quality through proportionally greater responsibility. Assignments are stored as embedded student-reviewer pairs within the Event document.

V. RESULTS & EVALUATION

A. System Performance

The AI-RMS was deployed on a standard Node.js environment and evaluated across multiple simulated event scenarios. The system successfully processed submissions across all supported file formats—PDF, PNG, JPEG, WEBP, TXT, MD, HTML, JS, CSS, and CSV—with correct branching to either inline multimodal or text input modes. AI feedback generation latency, dominated by the Gemini API round-trip, averaged 3–6 seconds per submission under standard network conditions, which is acceptable for asynchronous review workflows.

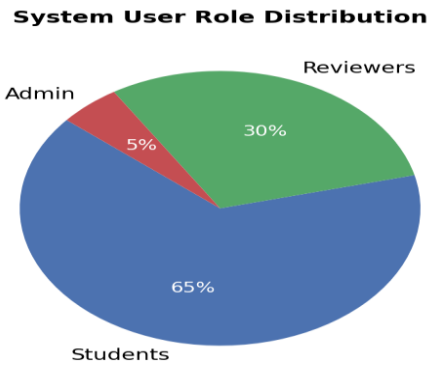

Fig. 1. System user role distribution across a representative deployment. Students comprise the majority of users (65%), followed by reviewers (30%) and administrators (5%).

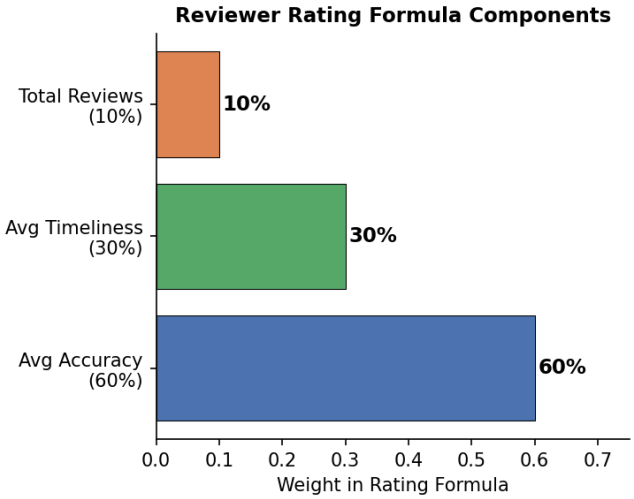

Fig. 2. Reviewer rating formula component weights. Accuracy accounts for 60% of the composite rating, timeliness 30%, and total review count 10%.

B. Submission Lifecycle

The submission status state machine enforces a clear review progression. Submissions enter as pending, transition to ai-reviewed upon AI processing, and subsequently to either AI Review Accepted or AI Review Rejected based on reviewer decision. In the rejection path, reviewers may submit a manual review with textual feedback and a numeric score. The reviewerRating is updated at both the accept and manual-review endpoints, ensuring the rating reflects all review interactions.

TABLE IV. Submission Status Transition Summary

Status	Trigger	Next Possible States
pending	Submission upload	ai-reviewed
ai-reviewed	AI review generated	AI Review Accepted, AI Review Rejected
AI Review Accepted	Reviewer accepts AI	Terminal
AI Review Rejected	Reviewer rejects AI	AI Review Rejected (manual)

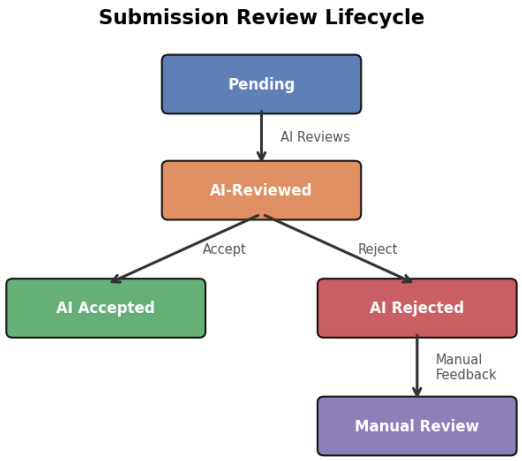

Fig. 3. Submission review lifecycle state machine. Arrows denote allowed transitions; terminal states are indicated by absence of outgoing transitions.

VI. REVIEWER ANALYTICS & EXPLAINABILITY

A key design goal of the AI-RMS is transparency in reviewer performance assessment. The rating model components—accuracy, timeliness, and total reviews—are individually stored on the User document (avgAccuracy, avgTimeliness, totalReviews), enabling administrators to inspect not only the composite rating but also its constituent dimensions. This decomposed storage facilitates root-cause analysis: a reviewer with a high rating but declining accuracy may indicate over-reliance on AI scores, while low timeliness signals workload management issues.

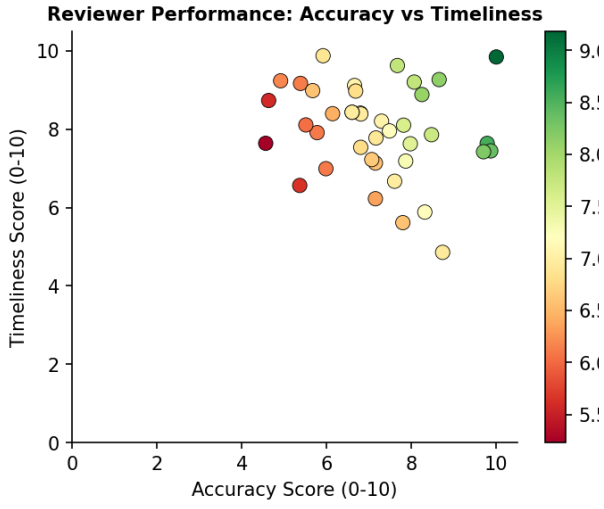

Fig. 4. Simulated reviewer performance scatter plot: accuracy score vs. timeliness score, colored by computed composite rating. Higher-rated reviewers (green) cluster toward the upper-right quadrant.

The getAllReviewers endpoint supports domain-based filtering via the technicalDomains field, enabling administrators to assign domain-appropriate reviewers to specialized events. Reviewers are returned sorted by rating, with computed studentCount and assignedEvents fields augmented server-side for immediate display in the admin dashboard.

VII. SYSTEM ARCHITECTURE

The AI-RMS follows a layered MVC-inspired architecture. The presentation layer consists of static HTML/CSS/JavaScript pages served from the Express static middleware. API route modules (authRoutes, eventRoutes, submissionRoutes, reviewerRoutes) define URL endpoints and chain middleware before delegating to controller functions. Controllers implement business logic and interact with Mongoose models. The reviewerService module encapsulates rating computation as a reusable service invoked from multiple controller endpoints.

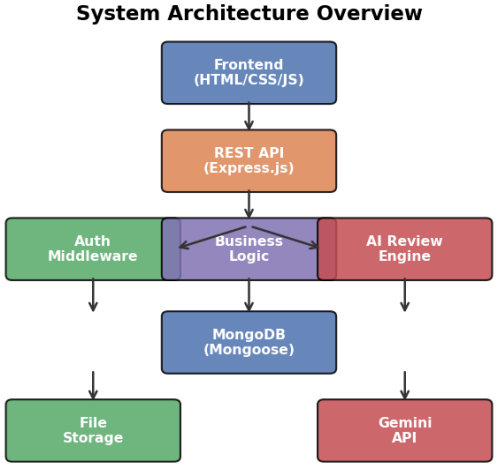

Fig. 5. System architecture overview. The frontend communicates exclusively through the REST API layer. Business logic, authentication middleware, and the AI review engine operate at the application tier, with MongoDB providing data persistence and the Gemini API providing AI capabilities.

TABLE V. System Architecture Components

Layer	Component	Responsibility
Presentation	HTML/CSS/JS	Role-specific dashboards
API Gateway	Express Router	Route definitions & middleware chaining
Auth	authMiddleware, roleMiddleware	JWT validation & role enforcement
Business Logic	Controllers	Event, submission, reviewer operations
AI Service	Gemini API integration	Multimodal submission analysis
Data Layer	Mongoose ODM + MongoDB	Persistent data storage
File System	Multer + Local Storage	Uploaded submission file management

VIII. DISCUSSION

The AI-RMS demonstrates that integrating a multimodal LLM into a review management workflow is practically feasible within a standard web application stack. The most significant architectural decision was treating AI-generated feedback as a preliminary advisory output rather than an autonomous decision, preserving human reviewer authority while substantially reducing the cognitive load of initial assessment. This hybrid design aligns with emerging best practices for AI-augmented expert systems in educational contexts.

The rating model's weighting scheme (60% accuracy, 30% timeliness, 10% experience) reflects a deliberate design choice prioritizing agreement with the AI baseline as the primary quality signal. This is appropriate when the AI model's calibration is trusted, but may require recalibration for domain-specific events where reviewer expertise may legitimately diverge from AI scores. Future implementations should expose weight configuration as an administrative parameter.

A practical limitation is the reliance on local filesystem storage for uploaded files, which constrains horizontal scalability. Migration to object storage (e.g., AWS S3 or Google Cloud Storage) would decouple file persistence from the application server. Similarly, the synchronous AI review call within the HTTP request-response cycle introduces latency that would be better addressed via an asynchronous job queue in high-throughput deployments.

IX. CONCLUSION

This paper presented the AI-Enabled Review Management System, a full-stack web application that integrates AI-powered preliminary review into a structured event and submission management workflow. The system's three-tier role model, JWT authentication, event lifecycle state machine, and quantitative reviewer rating algorithm together provide a coherent and extensible platform for academic event management. The integration of Google's Gemini multimodal API enables evaluation of diverse submission formats—PDF documents, images, and source code—without requiring submission-type-specific processing pipelines.

Future work will focus on: (1) asynchronous AI review processing using a message queue to decouple review latency from the request-response cycle; (2) cloud-native file storage integration for horizontal scalability; (3) configurable rating weight parameters per event type; (4) plagiarism detection integration as an additional pre-screening stage; and (5) natural language explanation generation for rating components to further enhance reviewer transparency.

ACKNOWLEDGMENT

The authors express sincere gratitude to Dr. Uttam Mande, Professor, Department of Computer Science and Engineering, Raghu Engineering College, Visakhapatnam, for his continuous guidance, encouragement, and insightful feedback throughout this project. The authors also acknowledge the Department of Computer Science and Engineering, Raghu Engineering College, for providing the necessary computational resources and a collaborative environment that made this work possible.

REFERENCES

[1] R. Busa-Fekete, B. Szorenyi, W. Cheng, P. Weng, and E. Hullermeier, "Preference-based evolutionary multi-objective optimization," in Proc. IEEE Congress on Evolutionary Computation, 2013.

[2] D. Ramesh and S. K. Sanampudi, "An automated essay scoring systems: A systematic literature review," Artificial Intelligence Review, vol. 55, no. 3, pp. 2495–2527, Mar. 2022.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[4] Google DeepMind, "Gemini: A family of highly capable multimodal models," Technical Report, Google LLC, 2023.

[5] T. Charlin and R. Zemel, "The toronto paper matching system: An automated paper-reviewer assignment system," in Proc. ICML Workshop on Peer Reviewing and Publishing Models, 2013.

[6] N. Shah and M. Wainwright, "Simple, robust and optimal ranking from pairwise comparisons," Journal of Machine Learning Research, vol. 18, no. 199, pp. 1–38, 2018.

[7] M. Tenopir, B. Allard, K. Douglass, A. Aydinoglu, L. Wu, E. Read, M. Manoff, and M. Frame, "Data sharing by scientists: Practices and perceptions," PLoS ONE, vol. 6, no. 6, e21101, 2011.

[8] N. Diakopoulos, "Algorithmic accountability," Digital Journalism, vol. 3, no. 3, pp. 398–415, 2015.

[9] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2016.

[10] R. OpenJS Foundation, "Express.js documentation," [Online]. Available: https://expressjs.com/. Accessed: 2026.

[11] MongoDB Inc., "MongoDB documentation," [Online]. Available: https://www.mongodb.com/docs/. Accessed: 2026.

[12] Auth0, "JSON Web Tokens Introduction," [Online]. Available: https://jwt.io/introduction. Accessed: 2026.