Prof. Tejaswini K, 2026, 14:5
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Design And Implementation Of An Ai-Assisted Final Year Project Management System With Rubric-Based Document Evaluation

Prof. Tejaswini K[1], Chiranth Rd[2], Utsav Kumar[3], Mohit Kumar[4]
Computer Science And Business System
BMS College Of Engineering

Abstract—The management of final-year academic projects involves proposal submission, supervisor coordination, document exchange, progress monitoring and academic evaluation. When these activities are distributed across separate communication channels, students and faculty may face fragmented records, delayed feedback and limited visibility into project progress. This paper presents the design and implementation of an AI-assisted Final Year Project Management System developed using the MongoDB, Express.js, React.js and Node.js (MERN) stack. The application provides role-specific functionality for students, teachers and administrators, including proposal management, document upload and download, feedback, resource sharing and real-time notifications using Socket.IO. The system also integrates the Grok application programming interface (API), provided by xAI, to support rubric-based academic document evaluation. Submitted documents are assessed against ten predefined criteria with assigned weightages that total 100 marks. The AI module generates criterion-wise marks and improvement suggestions, and students are allowed two AI-assisted evaluation attempts before final submission. The final assessment remains subject to teacher review, and teachers can modify AI-generated marks before finalizing the grade. The paper describes the system architecture, implementation workflow and assessment design. Quantitative validation of grading agreement, scoring reliability and system performance is identified as necessary future evaluation; no such metrics are claimed in this manuscript.

Keywords—Artificial intelligence, final year project management, MERN stack, large language models, rubric-based assessment, academic document evaluation, role-based access control, human-in-the-loop grading.

I. INTRODUCTION

Final-year projects are a significant component of undergraduate education, particularly in engineering and computer science programmes. They allow students to apply theoretical knowledge, develop practical skills and demonstrate their ability to solve real-world problems. Managing these projects requires coordination among students, faculty supervisors and administrative personnel throughout the academic lifecycle.

In many academic environments, project-related activities are handled through a combination of email, messaging applications, shared folders and manual records. Although these tools support individual tasks, they do not necessarily provide a unified view of proposal status, document submissions, supervisor feedback and project progress. Students may have difficulty identifying the latest feedback or tracking the state of an approval, while teachers may need to coordinate several groups and maintain records across different channels.

Web-based project management systems can centralize these activities by providing shared workspaces, role-specific access and structured communication. Prior work on final-year project supervision and e-collaboration has highlighted the value of online tools for managing student-supervisor interactions, project records and progress monitoring [1], [2]. At the same time, large language models (LLMs) have created new opportunities for automated document analysis, scoring and feedback generation. Research on LLM-assisted assessment indicates potential value as a support mechanism, while emphasizing the need to examine reliability and retain human oversight [3], [4].

This paper presents a Final Year Project Management System that combines project administration and

© 2026 Prof. Tejaswini K, This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly credited.

supervision workflows with an AI-assisted document evaluation module. The core application is developed using the MERN stack. The AI module communicates with the Grok API to evaluate submitted academic documents against a predefined rubric, produce criterion-wise marks and suggest improvements. Students can use the AI evaluation facility twice before final submission. Teachers review the final submission and can modify the AI-generated marks before finalizing the assessment.

The main contributions of the implementation are:

• Centralized management of project proposals, submissions, feedback and academic resources.

• Role-based interfaces and backend authorization for students, teachers and administrators.

• Document upload, download and metadata management using Multer and MongoDB.

• Real-time event notifications using Socket.IO, with notification records persisted in MongoDB.

• Integration of the Grok API for rubric-based document evaluation and improvement suggestions.

• A two-attempt AI evaluation workflow followed by teacher-controlled final grading.

II. LITERATURE REVIEW

1. Digital Management Of Final-Year Projects

Bakar et al. described a web-based supervision management system intended to support student-supervisor coordination and project monitoring. The work included student and supervisor profiles, meeting management and progress monitoring, illustrating how centralized systems can support academic project supervision [1]. Lounas et al. presented an e-collaboration application for final-year project management that addressed communication, coordination, resource sharing and evaluation among students, advisors and administrative staff [2]. These studies provide a basis for organizing academic project activities through a shared web application.

2. Large Language Models For Assessment

Mizumoto and Eguchi investigated the use of a language model for automated essay scoring and compared generated scores with benchmark assessments. Their work considered the reliability and accuracy of language-model-based scoring and discussed its potential to support human evaluation [3]. Xiao et al. examined human-AI collaborative essay scoring and proposed a framework in which LLM-generated scores and feedback are used alongside human assessment [4]. This line of research is relevant to systems that use AI to provide preliminary document evaluation while retaining instructor authority over final grades.

3. Positioning Of The Proposed System

The proposed system does not introduce a new language model or claim a novel scoring algorithm. Its contribution is the integration of a rubric-based LLM assessment workflow into a role-based academic project management application. The combination includes two student evaluation opportunities, criterion-wise feedback and teacher review of the final submission. The effectiveness of the scoring component must be established through comparison with human evaluators; therefore, the present paper describes the implemented workflow without claiming measured grading accuracy.

III. PROBLEM DEFINITION AND OBJECTIVES

1. Problem Definition

The project lifecycle includes proposal preparation, supervisor coordination, document submission, feedback, revision and final assessment. When these activities are distributed across separate tools, academic records can become fragmented and participants may lack a consistent view of project status. In addition, preliminary assessment of lengthy reports against multiple criteria can be repetitive. The problem addressed by this work is the design of a centralized application that supports the project lifecycle and provides AI-assisted document feedback without removing the teacher from final grading.

2. Objectives

• Provide a centralized web platform for project proposal submission and supervision.

• Separate student, teacher and administrator capabilities through role-based access control.

• Support document upload, download, feedback and teacher-provided resources.

• Provide real-time notifications for relevant project events.

• Integrate an LLM API to assess academic documents against ten weighted criteria.

• Provide criterion-wise marks and improvement suggestions, with two AI evaluation attempts.

• Allow teachers to review and modify AI-generated scores before finalizing grades.

IV. SYSTEM ARCHITECTURE

The application follows a client-server architecture. The React frontend provides role-specific dashboards and communicates with the Express backend through application routes. The backend performs authentication, authorization, project workflow operations, file handling and database access. MongoDB stores application records. Socket.IO provides real-time event delivery. For AI-assisted assessment, the backend communicates with the external Grok API using an API key.

The high-level workflow is represented as follows:

User Interface → Express/Node.js Backend → MongoDB
↘ Multer File Storage
↘ Socket.IO Notifications
↘ Grok API → Scores and Feedback

The principal components are summarized in Table I.

Component	Technology / Responsibility
Presentation layer	React.js dashboards for Student, Teacher and Admin roles
Application layer	Node.js and Express.js routes, controllers and middleware
Data layer	MongoDB with Mongoose models for users, projects, proposals, submissions, feedback and notifications
File handling	Multer processes uploads; files are stored in a backend directory and metadata/paths are stored in MongoDB
Real-time layer	Socket.IO delivers event notifications; notification records are persisted in MongoDB
AI assessment layer	Grok API is called through an API key to generate rubric-based scores and improvement suggestions

1. Technology Stack

Layer	Technology	Purpose
Frontend	React.js	Role-specific user interfaces and dashboards
Backend	Node.js, Express.js	Server-side application logic and REST endpoints
Database	MongoDB, Mongoose	Persistence of application records
Authentication	JWT, bcrypt	Token-based authentication and password hashing
File management	Multer, local file storage	Processing and storing uploaded documents
Real-time events	Socket.IO	Live notification delivery
AI integration	Grok API (xAI)	Document analysis, scoring and feedback

2. Role-Based Access Control

The system defines three principal roles. Students can submit proposals, upload documents, view project status, access feedback, download resources and receive notifications. Teachers can review assigned project work, provide feedback, manage submissions, share resources and review AI-generated evaluations. Administrators manage student and teacher accounts and oversee administrative functions, including group management. Backend middleware checks the authenticated user role before protected operations are executed. This provides server-side authorization in addition to role-specific frontend navigation.

3. System Implementation

4. Authentication And Authorization

The application uses JSON Web Tokens (JWT) for authentication. Passwords are hashed using bcrypt before being stored. Protected backend routes use authentication and role-checking middleware to determine whether a request is permitted. The current implementation stores JWTs in browser local storage. This is an implementation detail with security implications, particularly in relation to cross-site scripting (XSS), and should be considered when preparing a production deployment.

5. Proposal And Supervision Workflow

Students submit project proposals through the application, including project-related details, team information and supervisor preferences. The proposal is then made available for the relevant review workflow. Status updates communicate whether a proposal is pending, approved, rejected or requires revision. Students can monitor proposal status through their dashboard. Teachers manage assigned student work and provide review feedback. The precise division of approval responsibility between administrative and teaching roles should be described according to the final deployed configuration.

6. Document Management

Multer is used to process document uploads. Uploaded files are stored in a designated directory within the backend project, while MongoDB stores the associated metadata and file paths. Teachers can access student submissions, provide feedback and upload supporting resources. Students can download teacher-provided materials and update or resubmit documents when revisions are required. The current functionality supports document replacement and resubmission; it is not described as full version control unless historical file versions are retained.

7. Real-Time Notifications

Socket.IO is integrated to deliver notifications for relevant events, such as proposal status changes, document submissions and feedback updates. Notification records are also stored in MongoDB and displayed through the dashboard notification panel. The application uses structured feedback and notifications for academic communication; it does not include a dedicated live-chat module.

V. AI-ASSISTED DOCUMENT EVALUATION

1. Grok Api Integration

The AI module uses an API-key-based connection to the Grok API provided by xAI. The external language model analyses submitted academic documents and returns evaluation information. The exact model identifier, API request configuration, prompt template, document extraction process and response schema must be confirmed from the implementation code. Accordingly, this paper identifies the service as the Grok API and does not assume a particular model version.

2. Rubric-Based Evaluation

The AI module evaluates documents against ten predefined academic criteria. The criteria cover standard report components, including formatting, introduction, objectives and methodology, together with other applicable content and presentation dimensions. Each criterion has an assigned weightage, and the maximum weightages sum to 100 marks. The system returns criterion-wise marks and a cumulative score out of 100.

Let w_i denote the maximum weightage assigned to criterion i and m_i denote the marks awarded for that criterion, where $0 \leq m_i \leq w_i$. The total score is:

$$S_AI = \Sigma(i=1 \text{ to } 10)\ m_i, \qquad \Sigma(i=1 \text{ to } 10)\ w_i = 100$$

The exact names and weightages of the ten criteria should be reproduced from the rubric configured in the application. They are not enumerated in this manuscript because the complete rubric has not yet been supplied.

3. Feedback And Revision Attempts

In addition to marks, the AI module generates improvement suggestions to help students identify weaknesses in their documents. Students can access the AI evaluation facility twice before final submission. This creates a limited revision cycle: the student submits a document, receives criterion-wise scores and suggestions, revises the document if necessary, and may request one further AI evaluation before proceeding to final submission.

4. Teacher Review And Final Grading

AI-generated marks are preliminary assessment outputs and do not automatically become the final academic grade. Following final submission, the teacher reviews the document and the AI-generated evaluation. The teacher can modify the score before finalizing the assessment. This human-in-the-loop design preserves the teacher's authority over the final academic decision and provides an opportunity to correct inappropriate or inconsistent AI outputs.

The workflow can be summarized as:

Document Upload → AI Evaluation → Marks + Suggestions
→ Revision / Second Attempt → Final Submission
→ Teacher Review and Score Modification → Final Grade

VI. DATA MANAGEMENT AND SECURITY CONSIDERATIONS

MongoDB maintains application records such as user accounts, project information, proposals, submissions, feedback and notifications. Uploaded documents are stored in the backend file directory, while associated metadata and paths are maintained in the database. The application uses password hashing and JWT-based authentication, together with role checks on protected backend operations.

The current local-development implementation requires further security hardening before institutional deployment. Areas for review include token storage, input validation, file type and size restrictions, secure file access, rate limiting, logging and dependency maintenance. Since submitted academic documents are transmitted to an external AI API for evaluation, the institution should also define appropriate rules for student notice and consent, confidentiality, retention and third-party processing. API keys must be stored in server-side environment configuration and must not be exposed in frontend code or public repositories.

VIII. IMPLEMENTATION VERIFICATION AND EVALUATION

A. Functional Test Plan

The following test cases identify the main functions that should be verified. The expected outcomes are test specifications and must not be interpreted as completed experimental results.

ID	Function	Expected outcome
T1	Authentication	Valid users can authenticate and access their role-specific dashboard.
T2	Role authorization	Protected operations reject users without the required role.
T3	Proposal submission	A submitted proposal is stored and its status is visible.
T4	Proposal review	Relevant reviewers can update proposal status.
T5	Document upload	The file is stored and its metadata is recorded.
T6	Document download	Authorized users can retrieve permitted files.
T7	Feedback	Teacher feedback is visible to the relevant student.
T8	Notifications	Relevant events generate stored and delivered notifications.
T9	AI evaluation	The AI returns criterion-wise marks and improvement suggestions.
T10	Attempt restriction	The student is limited to two AI evaluation attempts.
T11	Teacher review	The teacher can review and modify AI-generated marks.
T12	Final grading	The teacher can finalize the assessment.

1. Quantitative Evaluation Required

The project information currently available does not include measured AI grading accuracy, agreement with teacher-assigned marks, API response time, document-processing time or concurrent-user capacity. These values are therefore not reported. To evaluate the AI assessment component, a future study should compare AI-generated scores with scores assigned independently by qualified teachers across a representative set of anonymized documents.

One possible measure is mean absolute error (MAE):

$$MAE = (1/n) \Sigma(i=1 \text{ to } n) |A_i - H_i|$$

where A_i is the AI-generated score, H_i is the corresponding teacher-assigned score and n is the number of evaluated documents. Additional analysis may examine criterion-level agreement, repeatability across repeated evaluations, API response time and the proportion of successful functional test cases. Such measurements are necessary before making claims about grading reliability, time savings or scalability.

VII. DISCUSSION

The implementation combines academic project administration and AI-assisted document evaluation within one application. Role-specific dashboards and backend authorization provide a structured environment for students, teachers and administrators. Proposal workflows, document exchange, feedback and notifications support the coordination activities that occur throughout the project lifecycle.

The AI module adds a formative assessment facility by providing criterion-wise marks and improvement suggestions. The two-attempt limit allows students to use the feedback for revision before final submission. Teacher review is an important control because LLM outputs may be inconsistent, may misinterpret technical content or may assign marks that do not fully reflect institutional expectations. The system therefore positions AI as an assessment-support mechanism rather than as an autonomous grading authority.

The contribution of the work is primarily architectural and integrative. MERN technologies, role-based access control, Socket.IO and external LLM APIs are established technologies. The paper does not claim that these technologies are individually novel. Instead, it describes their integration into a specific academic workflow. The strength of the research contribution will depend on documenting the implementation accurately and validating the AI assessment process with real evaluation data.

VIII. LIMITATIONS AND FUTURE WORK

1. Limitations

• The exact Grok model identifier and API request configuration have not yet been confirmed.

• The complete list of ten rubric criteria and their individual weightages must be documented from the implemented configuration.

• No quantitative comparison between AI-generated marks and teacher-assigned marks has been provided.

• No measured response-time, reliability, scalability or concurrent-user results are currently available.

• The application is described as a local development implementation; production deployment has not been established.

• The application uses JWTs stored in browser local storage, which requires security review.

• The use of an external API introduces service availability, cost and student-document privacy considerations.

• The implementation incorporates an instructional template/tutorial; reused components and the team's original modifications must be clearly acknowledged.

2. Future Work

• Conduct a rubric-level comparison between AI scores and independent faculty assessments.

• Measure scoring repeatability, response time and functional reliability.

• Record the exact model identifier, prompt configuration and document preprocessing pipeline.

• Introduce assessment history and traceability for teacher modifications.

• Improve secure file handling, token protection and production deployment practices.

• Evaluate usability with students and faculty under an approved institutional process.

• Extend the evaluation workflow to support additional rubrics for different academic disciplines.

IX. CONCLUSION

This paper presented the design and implementation of an AI-assisted Final Year Project Management System developed using the MERN stack. The application provides a centralized environment for project proposals, supervision, document exchange, feedback and real-time notifications through role-

based functionality for students, teachers and administrators.

The integration of the Grok API introduces rubric-based academic document evaluation. The AI module evaluates submitted documents across ten weighted criteria, generates marks out of 100 and provides improvement suggestions. Students are permitted two AI-assisted evaluation attempts before final submission. Teachers retain authority over the final assessment and can review and modify AI-generated marks.

The system provides an integrated implementation of academic project management and AI-assisted formative assessment. Its effectiveness as a grading-support tool remains to be established through systematic comparison with human evaluators and further functional and performance testing. Future work will focus on validation, security hardening and deployment readiness.

REFERENCES

1. M. A. Bakar, N. Jailani, Z. Shukur, and N. F. M. Yatim, "Final Year Supervision Management System as a Tool for Monitoring Computer Science Projects," Procedia - Social and Behavioral Sciences, vol. 18, pp. 273–281, 2011, doi: 10.1016/j.sbspro.2011.05.039.
2. R. Lounas, I. Hamzaoui, N. Bouguelmouna, and H. Mokrani, "An E-Collaboration Application for Final-Year Project Management," International Journal of e-Collaboration, vol. 19, no. 1, pp. 1–17, 2023, doi: 10.4018/IJeC.315787.
3. A. Mizumoto and M. Eguchi, "Exploring the potential of using an AI language model for automated essay scoring," Research Methods in Applied Linguistics, vol. 2, no. 2, Art. no. 100050, 2023, doi: 10.1016/j.rmal.2023.100050.
4. C. Xiao, W. Ma, Q. Song, S. X. Xu, K. Zhang, Y. Wang, and Q. Fu, "Human-AI Collaborative Essay Scoring: A Dual-Process Framework with LLMs," in Proceedings of the 15th International Learning Analytics and Knowledge Conference (LAK '25), 2025, pp. 293–305, doi: 10.1145/3706468.3706507.
5. xAI, "Grok API Documentation," [Online]. Available: https://docs.x.ai/. Accessed: Nov. 9, 2026.
6. MongoDB, "MongoDB Manual," [Online]. Available: https://www.mongodb.com/docs/. Accessed: Nov. 9, 2026.
7. OpenJS Foundation, "Express - Node.js web application framework," [Online]. Available: https://expressjs.com/. Accessed: Nov. 9, 2026.
8. Socket.IO, "Socket.IO Documentation," [Online]. Available: https://socket.io/docs/v4/. Accessed: Nov. 9, 2026.
9. Khaleel Khan Mohammed. "A Survey on DigitalHealth Care Data Analysis Techniques forDeveloping Machine Learning Models