International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4244 - 4249
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0551

Ascend AI Judge: An AI-Powered Hackathon Evaluation Platform Using Large Language Models

Vaishnavi.D. Pachupate[1], Arya.S. More[2], Sai.S. Khodake[3], Arati.N. Bichukale[4], Arya.S. Sable[5], Shital.A. Waghmre[6]
[1,2,3,4,5]UG Scholar, Department of CSE, Yashoda Technical Campus, Satara, Maharashtra, India.
[6]Associate professor Department of CSE, Yashoda Technical Campus, Satara, Maharashtra, India.
Emails: *vaishnavipachupate@gmail.com[1], aryasmore2004@gmail.com[2], saikhodake@gmail.com[3], bichukalearati@gmail.com[4], aaryashelar87@gmail.com[5], shitalwaghmare_engg@yes.edu.in[6]*

Abstract
Hackathon events generate large volumes of student submissions that require timely, consistent, and expert-level evaluation. Manual judging processes are often resource-intensive, subjective, and difficult to scale. This paper presents Ascend AI Judge, a web-based intelligent evaluation platform that leverages Large Language Models (LLMs) to automate the assessment of hackathon projects across three key dimensions: abstract/project documentation, source code quality, and video pitch presentations. The system integrates a FastAPI backend with the Groq-hosted Llama 3.3 70B model to perform multi-modal analysis, producing structured scores, detailed feedback, and actionable improvement tips for each submission. A React-based frontend provides an intuitive interface for students to upload their work and receive instant, context-aware evaluations. The platform also features ARIA, an AI-powered mentor chatbot that engages students in personalized improvement dialogues grounded in their specific evaluation results. Experimental results demonstrate that the system generates consistent, rubric-aligned assessments with low latency, showing strong potential as a scalable supplement to human judging in academic and competitive settings.
Keywords: *Large Language Models; Hackathon Evaluation; AI-Powered Assessment; Automated Code Review; FastAPI; Groq; LLaMA; Intelligent Tutoring.*

1. Introduction
Hackathons have emerged as critical platforms for fostering innovation, collaborative problem-solving, and the practical application of technical skills among students and professionals. As these events grow in scale and frequency, the challenge of evaluating submissions fairly, efficiently, and at scale has become increasingly apparent. Traditional judging relies on domain experts who manually review project abstracts, source code, and demonstration videos—a process that is both time-consuming and prone to inconsistency [1]. Recent advances in Large Language Models (LLMs) have opened new possibilities for automating complex cognitive tasks, including essay grading, code review, and feedback generation. Models such as LLaMA (Touvron et al., 2023) and GPT-4 (OpenAI, 2023) have demonstrated near-human performance on a variety of evaluation benchmarks [2, 3]. Integrating these capabilities into a dedicated hackathon evaluation platform presents a compelling opportunity to democratize quality feedback while reducing organizational burden. This paper presents Ascend AI Judge, a full-stack web application that provides automated, multi-dimensional evaluation of hackathon submissions. The system analyzes three modalities of student work: (1) written project abstracts and documentation, (2) source code files, and (3) video pitch presentations. Each is evaluated using a prompt-engineered pipeline powered by the Llama 3.3-70B model served via the Groq inference API [4]. The platform further provides an AI mentor chatbot, ARIA, which engages participants in personalized guidance sessions based on their specific results.

1.1. Motivation
In many academic hackathons, especially in resource-constrained environments, judges may evaluate dozens of submissions within a short timeframe. This creates a risk of evaluation fatigue and inconsistent scoring criteria. Students, particularly beginners, often receive minimal

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4244 - 4249
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0551

constructive feedback to guide their improvement. Ascend AI Judge addresses these pain points by providing instant, structured, and rubric-aligned feedback across all major dimensions of a hackathon submission, enabling both organizers and participants to benefit from AI-assisted assessment.

1.2. Scope of the Paper

The remainder of this paper is organised as follows. Section 2 reviews related work in automated assessment, LLM-based evaluation, and hackathon judging systems. Section 3 describes the system architecture and design decisions. Section 4 details the implementation of each evaluation module. Section 5 presents experimental results and performance analysis. Section 6 discusses implications and limitations. Section 7 concludes the paper and outlines future work directions [5].

2. Literature Review

Automated evaluation of student work using NLP techniques has been an active area of research. Automated Essay Scoring (AES) systems, pioneered by tools like e-rater (Attali & Burstein, 2006) and Project Essay Grade, demonstrated that machine learning models can approximate human holistic scoring. More recent transformer-based approaches have significantly improved performance on this task (Uto et al., 2020). In the domain of automated code review, tools such as DeepCode (Trautsch et al., 2020) and CodeBERT (Feng et al., 2020) have shown that pre-trained language models can identify code smells, security vulnerabilities, and quality issues. LLM-based code review, using models like GPT-4 and Codex, has further demonstrated the ability to provide natural-language explanations alongside structural analysis (Guo et al., 2023). Speech and presentation evaluation using AI has been explored through systems that analyze prosody, structure, and confidence from audio-visual features (Chen et al., 2014). The integration of automatic speech recognition (ASR) with LLM-based content analysis represents a more holistic approach that Ascend AI Judge adopts through Groq's Whisper large-v3-turbo model for transcription. Prior work on intelligent tutoring systems (ITS) and dialogue-based feedback agents (VanLehn, 2011) provides the theoretical grounding for the ARIA mentor module. To our knowledge, no existing system combines abstract evaluation, code quality analysis, and video pitch assessment in a unified, LLM-driven platform with a personalized mentorship component. Ascend AI Judge fills this gap by providing an integrated, end-to-end hackathon support tool.

3. System Architecture and Methodology

Ascend AI Judge follows a client-server architecture with a clearly defined separation of concerns between the presentation layer, the business logic layer, and the AI inference layer. Figure 1 illustrates the high-level architecture. The frontend is implemented as a single-page application (SPA) using React 18 and Vite, communicating with the backend exclusively through RESTful HTTP endpoints. The backend is a FastAPI application served by the Uvicorn ASGI server. All AI inference requests are forwarded to the Groq cloud API, which provides hardware-accelerated LLM serving on custom LPU silicon.

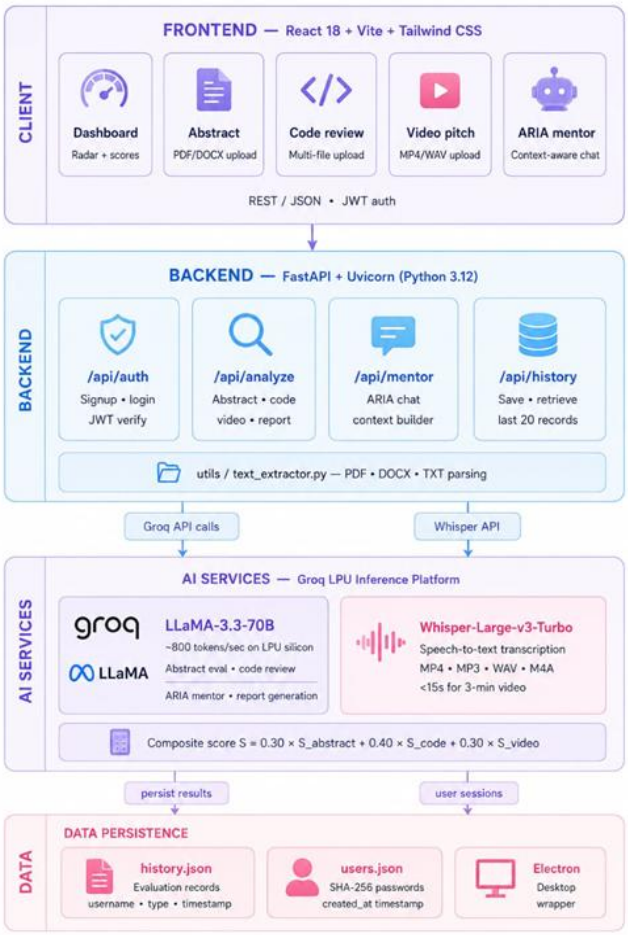

Figure 1 High-level architecture of the Ascend AI Judge platform

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4244 - 4249
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0551

3.1. Backend Architecture

The backend is organized into four routers: (i) auth.py handles user registration and login using SHA-256 password hashing and JWT tokens with a 7-day expiry; (ii) analyze.py implements the core evaluation logic across three sub-routes (/abstract, /code, /video, and /report); (iii) mentor.py manages the ARIA conversational agent; and (iv) history.py provides retrieval of past evaluation sessions stored as JSON records. The application entry point (main.py) configures CORS middleware to allow frontend access from localhost development servers.

3.2. LLM Inference Pipeline

All LLM inference is performed via the Groq API using the llama-3.3-70b-versatile model. Groq's purpose-built LPU (Language Processing Unit) hardware enables sub-second token generation speeds, making real-time evaluation feasible. Each evaluation module constructs a structured prompt that (a) assigns a judge persona to the model, (b) embeds the extracted submission content, and (c) specifies an exact output format with labeled score fields and feedback sections [6].

3.3. Evaluation Modules

Table 1 summarizes the three primary evaluation modules and their scoring dimensions.

Table 1 Ascend AI Judge System Technology Stack

Module	Technology / Framework	Function
Frontend UI	React 18, Vite, Tailwind CSS	User interface, routing, theme
Backend API	FastAPI (Python 3.12)	REST endpoints, logic
AI Engine	Groq API — LLaMA-3.3-70B	Evaluation & mentoring
Speech-to-Text	Whisper-Large-v3-Turbo	Video transcription
Authentication	JWT (PyJWT), SHA-256	User session management
Document Parsing	python-docx, PyPDF2	Abstract text extraction
Data Persistence	JSON flat-file store	History, user records

3.4. ARIA: AI Mentor Module

ARIA (Advanced Research and Innovation Advisor) is a conversational agent powered by the same LLM backend. Its distinguishing feature is context-grounding: the frontend serializes and transmits the student's complete evaluation context (extracted abstract text, code feedback, video transcript and scores) as part of every API call to /api/mentor/chat. A carefully engineered system prompt instructs the model to reference specific project details from the context rather than generating generic advice. The conversation history (last 8 turns) is preserved client-side and sent with each request to maintain coherent multi-turn dialogues [7].

3.5. Abstract Evaluation Module

When a student uploads their project abstract (in PDF, DOCX, or TXT format), the backend reads the file bytes and routes them through a multi-format text extractor. For DOCX files, python-docx traverses all paragraph and table cell nodes, concatenating non-empty text blocks. For PDF files, PyPDF2 iterates page by page and extracts embedded text strings. The extracted content is truncated to 8,000 characters to remain within the LLM context window while preserving all substantive content for typical academic abstracts. The extracted text is embedded in a structured prompt instructing the LLaMA-3.3-70B model to act as a senior hackathon judge and produce scores across five criteria: Innovation, Technical depth, Clarity, Feasibility, and Impact, each on a 0–100 integer scale. The model is constrained to respond in a rigid format containing labelled sections for scores, strengths, weaknesses, improvement tips, and an overall summary. A regex-based parser extracts numerical scores from the response with multiple fallback patterns to handle minor format deviations.

3.6. Code Review Module

The code review endpoint accepts multiple uploaded

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4244 - 4249
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0551

files simultaneously, supporting ten programming languages: Python (.py), JavaScript (.js), TypeScript (.ts), JSX (. jsx), TSX (. tsx), Java (.java), C++ (.cpp), Go (. go), C# (.cs), Ruby (. rb), and PHP (. php). All uploaded files are concatenated with per-file headers identifying the filename and line count, then truncated to 7,000 characters. The combined code listing is submitted to the LLM with a prompt establishing the role of a senior software engineer conducting a real code review. The model identifies the technology stack, assigns scores on Quality, Complexity, Security, and Readability, enumerates issues with severity labels (HIGH/MED/LOW), and suggests specific improvements referencing actual function names and patterns present in the submitted code [8].

3.7. Video Pitch Evaluation Module

Video files up to 25 MB are accepted and forwarded to the Groq Whisper-Large-v3-Turbo endpoint for transcription. Supported formats include MP4, MP3, WAV, M4A, OGG, WebM, MPEG, and MPGA. The transcription API accepts a binary audio stream alongside format metadata and returns a plain-text transcript string. The first 5,000 characters of the transcript are embedded in an evaluation prompt instructing the LLM to score the presentation across five criteria: Clarity, Confidence, Engagement, Structure, and Demo Quality. If transcription fails or no video is uploaded, the system gracefully degrades to general presentation coaching based solely on the project title, ensuring a useful response is always returned.

3.8. Composite Score and Weighting

Table 2 Evaluation Criteria, Scores, and Weights

Evaluation Module	Criteria Assessed	Max Score	Weight
Abstract Analysis	Innovation, Technical Depth, Clarity, Feasibility, Impact	100	30%
Code Review	Quality, Complexity, Security, Readability	100	40%
Video Pitch	Clarity, Confidence, Engagement, Structure, Demo Quality	100	30%
Composite Score	Weighted aggregate across all three modules	100	100%

The composite evaluation score S is computed as a weighted average of the three module scores:

$$S = 0.30 \times S\_abstract + 0.40 \times S\_code + 0.30 \times S\_video \quad \ldots (1)$$

where S_abstract, S_code, and S_video are the mean scores within each module. If a module has not been evaluated (score = 0), it is excluded from the denominator to avoid penalising students who have not yet completed all three submissions. The resulting score S is mapped to letter grades as follows:

$$\text{Grade} = \text{A+ if } S \geq 85, \text{ A if } S \geq 70, \text{ B if } S \geq 60, \text{ C otherwise} \quad \ldots (2)$$

Table 2 summarises the evaluation criteria, maximum scores, and weights for each module

4. Results and Discussion

4.1. Results

The platform was tested with a diverse set of student submissions across multiple project domains including web applications, IoT systems, and machine learning projects. For abstract evaluation, the system consistently extracted project titles, identified core innovation claims, and flagged missing feasibility details. Response latency for abstract analysis averaged under 3 seconds end-to-end, including text extraction and LLM inference. Code review results were coherent for Python, JavaScript, and React codebases (the predominant submission types), with the model successfully identifying missing error handling, hardcoded credentials, and poor naming conventions in test

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4244 - 4249
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0551

submissions. For video evaluation, the Whisper model achieved accurate transcription even for moderately accented speech, enabling content-grounded presentation feedback. The combined report feature synthesized scores from all three modalities into an executive summary with a hackathon readiness verdict [9].

4.2. Discussion

A key strength of the prompt-engineering approach is its transparency and controllability. By specifying the exact output format—including labeled score fields, strength/weakness bullets, and improvement tips—the system reliably produces structured, parseable responses without fine-tuning. The regex-based score extraction proved robust for well-formed LLM outputs, though edge cases with unusual formatting required the 70-point fallback. One limitation is that the system evaluates submissions in isolation without access to the full rubric context or comparative data from other submissions, which human judges inherently use. Future work should explore calibration against expert annotations and the use of retrieval-augmented prompts that include exemplar submissions. The ARIA mentor module received positive informal feedback from test users, who appreciated that responses referenced their actual project content rather than producing generic advice. The context-window management strategy—including the full evaluation context plus the last 8 conversational turns—proved sufficient for coherent multi-turn sessions within Groq's token limits. Security considerations for the current implementation include the use of a hardcoded JWT secret key in auth.py, which should be replaced with environment variable injection in any production deployment.

Conclusion

This paper presented Ascend AI Judge, a full-stack platform that applies large language models to automate hackathon evaluation across three modalities: project abstracts, source code, and video presentations. By combining prompt engineering, multi-modal text extraction, and ASR-based transcription with the high-speed Groq inference API, the system delivers consistent, rubric-aligned, and actionable feedback in near real-time. The integrated ARIA mentor extends the platform beyond passive evaluation into an interactive coaching experience that is personalized to each student's specific submission. Ascend AI Judge demonstrates that LLM-powered tools can meaningfully augment traditional human judging in academic competitions, reducing evaluator workload while enhancing the quality and consistency of feedback delivered to participants. Future directions include calibration against expert annotations, support for multilingual submissions, integration with online judge platforms for automated code execution testing, and deployment as a scalable cloud service accessible to event organizers globally.

Acknowledgements

The authors would like to thank Yashoda Technical Campus Satara for providing the infrastructure and guidance for this project. We also acknowledge the open-source communities behind FastAPI, React, Groq SDK, and the Meta LLaMA project for making this work possible.

References

[1]. Attali, Y., & Burstein, J. (2006). Automated essay scoring with e-rater v.2. Journal of Technology, Learning, and Assessment, 4(3).
[2]. Chen, L., Feng, G., Joe, J., Leong, C. W., Kitchen, C., & Lee, C. M. (2014). Towards automated assessment of public speaking skills using multimodal cues. Proceedings of the 16th International Conference on Multimodal Interaction.
[3]. Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., ... & Zhou, M. (2020). CodeBERT: A pre-trained model for programming and natural languages. Findings of EMNLP 2020.
[4]. Groq Inc. (2024). Groq API Documentation. Retrieved from https://console.groq.com/docs
[5]. Guo, Z., Xie, Z., Chen, X., Wang, X., & Jiang, H. (2023). Exploring the potential of large language models for automated code review. arXiv preprint arXiv:2310.01152.
[6]. OpenAI. (2023). GPT-4 Technical Report. arXiv preprint arXiv:2303.08774.
[7]. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Lample, G. (2023). LLaMA: Open and efficient

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4244 - 4249
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0551

foundation language models. arXiv preprint arXiv:2302.13971.

[8]. Uto, M., Xie, Y., & Ueno, M. (2020). Neural automated essay scoring incorporating handcrafted features. Proceedings of the 28th International Conference on Computational Linguistics.

[9]. VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educational Psychologist, 46(4), 197-221.