

AbhyasAI: A Secure AI-Assisted Technical Assessment Platform for Controlled Remote Evaluation and Personalized Performance Feedback

Anshika Singh¹, Saurabh Yadav², Neha Singh³

¹Student, Department of Computer Science and Engineering, R.R. Institute Of Moderna Technology, Lucknow, India

²Student, Department of CS(AIML), R.R. Institute Of Moderna Technology, Lucknow, India

³Head of Department, Department of Computer Science and Engineering, R.R. Institute Of Moderna Technology, Lucknow, India

Corresponding author: anshusingh262005@gmail.com

Abstract - AbhyasAI is a web-based technical assessment platform designed to provide a controlled environment for evaluating candidates through structured tests while generating AI-assisted performance feedback. The system combines a React-based frontend, a Node.js and Express backend, MongoDB persistence, browser-level assessment controls, camera-enabled monitoring, and Azure AI Foundry integration for automated result analysis and report generation. Following an assessment, the platform analyzes the test outcome and generates a structured report covering the overall assessment, strengths, weaknesses, areas for improvement, and actionable recommendations. The frontend is deployed on Vercel and the backend is deployed on Render, with the application exposed through the AbhyasAI domain. Security validation was performed by testing the principal assessment-integrity scenarios implemented by the platform, including unauthorized API access, authentication manipulation, cross-user result access, duplicate submissions, direct API submission, tab switching, fullscreen exit, camera denial, refresh during an active assessment, and related access-control cases. The validated implementation provides a practical foundation for secure remote technical assessment and evidence-grounded AI feedback. This paper presents the system architecture, technology stack, security model, AI-feedback workflow, deployment, validation methodology, limitations, and directions for future empirical evaluation.

Index Terms - technical assessment, secure assessment environment, remote evaluation, AI-assisted feedback, Azure AI Foundry, web application, assessment integrity, personalized performance analysis.

Note: This manuscript is an implementation-grounded IEEE-style submission draft. Numerical performance claims have not been fabricated. These security results reported below refer to functional validation performed by the developer; controlled participant experiments and statistical evaluation remain recommended before journal submission.

I. INTRODUCTION

Online technical assessment has become important in recruitment, campus placement, professional certification, and skill development. A scalable assessment platform must do more than render questions: it must authenticate users, enforce access rules, preserve attempt state, evaluate answers consistently, protect administrative functions, and provide useful feedback after submission.

AbhyasAI addresses this problem as a full-stack system. Its public platform describes AI-powered exam preparation, institutional technical interviews, timed simulations, DSA practice, AI mock interviews, performance analytics, and topic-oriented preparation [1]. The submitted source repositories provide implementation evidence for the assessment, authentication, administration, database, AI analysis, and controlled test-environment layers.

The research contribution proposed here is an architecture in which assessment evidence is generated deterministically by the application and database, while generative AI is used only after submission to interpret that evidence. This distinction is important because a language model can generate fluent but unsupported statements. A trustworthy assessment system should therefore keep the source of truth outside the generative model.

The paper makes five contributions: (1) an implementation-grounded full-stack architecture; (2) a secure assessment workflow with authenticated sessions, attempt constraints, server-side scoring, and browser-level controls; (3) an AI feedback pipeline using Azure AI Foundry with structured instructions; (4) an analysis of privacy, security, and responsible-AI boundaries; and (5) an empirical methodology for evaluating the platform rather than relying on product claims.

II. RELATED WORK AND RESEARCH GAP

Research on online assessment includes automated grading, learning analytics, remote proctoring, and intelligent tutoring. Automated grading provides consistent correctness checks for objective questions, but a numerical score alone does not necessarily explain why a candidate struggled or what should be learned next.

Digital proctoring research has identified authentication, cheating detection, adoption, privacy, technical reliability, fairness, and candidate experience as recurring concerns [2], [3]. This is relevant to AbhyasAI because the platform uses browser camera access and recording as part of a controlled assessment environment. These mechanisms should be interpreted as integrity controls rather than complete proof of examination honesty.

Large language model research in education reports potential benefits in personalization and immediate feedback, but also identifies hallucination, bias, privacy, over-reliance, and inconsistent quality [4], [5]. The feedback framework of Hattie and Timperley emphasizes that feedback is valuable when it helps a learner understand current performance and decide what to do next [6].

The research gap addressed by AbhyasAI is the integration of these concerns in a technical assessment workflow: secure identity and assessment execution, objective server-side evaluation, and evidence-grounded generative feedback are implemented as connected but separable layers.

III. FULL-STACK TECHNOLOGY STACK

Layer	Technology	Role
Frontend	React 19, Vite	Candidate, assessment, result, and administration interfaces.
UI/Routing	React Router, Tailwind CSS, Bootstrap, Framer Motion	Navigation and presentation.
Backend	Node.js, Express 5	REST APIs, middleware, routing, assessment logic.
Database	MongoDB, Mongoose	Users, tests, questions, attempts, scores, feedback.
Authentication	JWT, HTTP-only cookies	Authenticated sessions and protected routes.
Password security	bcryptjs	Password hashing before storage.
AI	Azure AI Foundry	API access for automated result analysis and report generation.
Media	MediaRecorder API	Browser camera evidence capture; audio disabled in analyzed implementation.
Email	Nodemailer / Gmail SMTP	OTP and account-related email workflows.
Files	Multer, ExcelJS/XLSX	Question import and report/export workflows.
OAuth	Google authentication library	Google authentication/verification workflow.
API docs	Swagger / OpenAPI	API documentation and testing support.
Deployment	Vercel + Render	Frontend and backend deployment respectively.

Table I. Technology stack and deployment environment of AbhyasAI.

IV. SYSTEM ARCHITECTURE

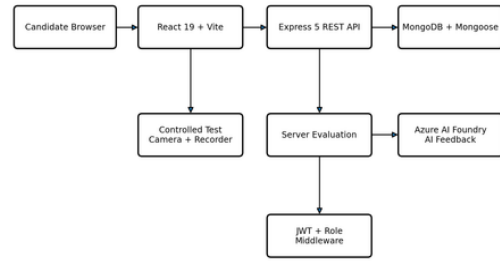


Fig. 1. Logical architecture of the deployed AbhyasAI platform.

The system follows a client-server architecture. The React frontend communicates with the Express backend through Axios. The backend uses Mongoose models to persist users, tests, questions, attempts, feedback, and related entities in MongoDB. Authentication is performed through JWTs stored in an HTTP-only cookie. The backend exposes separate route groups for user operations, student operations, administrator operations, and session verification.

The architecture contains two distinct assessment paths. The candidate-facing path manages test interaction and local browser controls. The server-facing path owns authoritative assessment data, question answers, score calculation, attempt state, and AI analysis. This division is particularly important for integrity because client-side values should not be trusted as the authoritative score.

V. AUTHENTICATION AND AUTHORIZATION

The backend implements user authentication through JWT verification. The userAuth middleware reads the refreshToken cookie, verifies the token using JWT_SECRET, retrieves the corresponding user from MongoDB while excluding the password, and attaches the user identifier and role to the request. The adminAuth middleware performs the same token validation and additionally requires the stored role to be admin.

The backend supports registration with OTP verification, email/password login, Google ID-token authentication, logout, password-reset workflows, and a /verifyme endpoint for validating the current session. Passwords are handled through bcryptjs rather than being stored in plaintext.

HTTP-only cookies provide an important security property because JavaScript running in the browser cannot directly read an HTTP-only cookie. Nevertheless, production security should also address cookie SameSite/Secure configuration, CSRF considerations, token lifetime and rotation, rate limiting, and secret management. These should be explicitly documented and tested before making strong security claims.

VI. DATA MODEL AND ASSESSMENT STATE

The MongoDB/Mongoose layer models the assessment domain explicitly. The Test model stores test name, year, question references, start and end times, creator, active status, category, duration, and total-question metadata. The Question model stores the question text, image reference, options, authoritative answer, topic, subject, difficulty level, explanatory metadata, and marks.

The AttemptTest model represents a candidate's attempt. It stores start and end timestamps, user and test references, attempt status, submitted answers, correct-answer count, score, AI analysis, and category. Most importantly, a compound unique index on userId and testId enforces the intended one-attempt-per-user-per-test structure at the database level. This is stronger than relying only on a frontend flag.

The question model includes subject and difficulty dimensions, allowing future research to analyze performance by topic and difficulty rather than only by total score. Such structure is valuable for personalized feedback and longitudinal learning analytics.

VII. SECURE TEST ENVIRONMENT

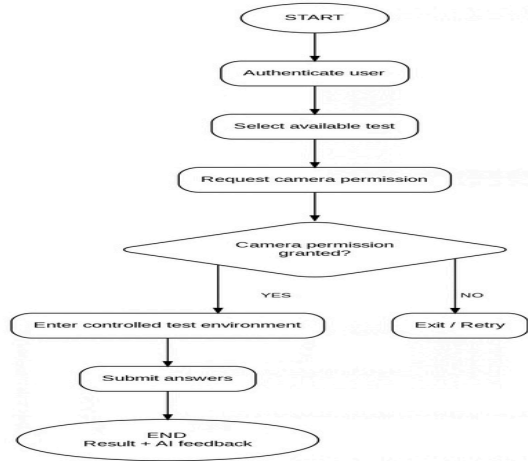


Fig. 2. Candidate assessment flow.

The frontend TestEnvironment component establishes a controlled browser session. Before starting the assessment, it requests camera access through `navigator.mediaDevices.getUserMedia`. The submitted implementation configures video capture and explicitly sets audio to false. Therefore, the current repository demonstrates camera capture and recording, not microphone recording. If microphone-enabled assessment is a required product feature, audio capture should be implemented and documented separately.

After test start, the frontend creates a `MediaRecorder` over the camera stream and collects recording chunks. On recording completion, the chunks are assembled into a video Blob and passed to a proof-storage service together with candidate metadata. The component also manages full-screen mode and tracks tab-switch events. Copy, cut, paste, and context-menu actions are restricted in the test interface.

These controls should be described as a layered integrity mechanism. Browser-level restrictions cannot prevent all forms of unauthorized assistance because a user may employ another device, operating-system capture, browser automation, or other external mechanisms. A research evaluation should therefore test the specific controls against a predefined threat model rather than claiming absolute cheating prevention.

VIII. SERVER-SIDE ASSESSMENT EVALUATION

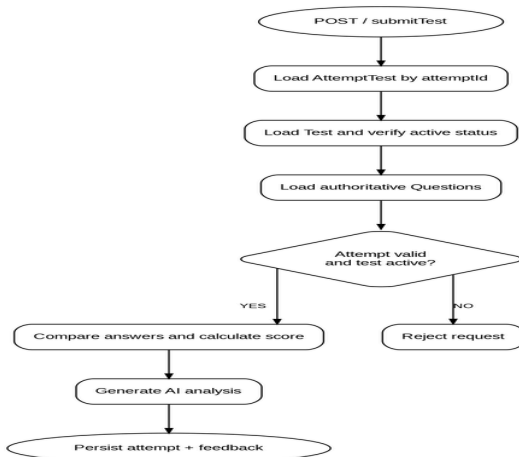


Fig. 3. Server-side submission and scoring flow.

The `submitTest` controller provides an important server-side trust boundary. It receives an `attemptId` and answers, retrieves the attempt, checks that the attempt exists, retrieves the corresponding test, and rejects submission if the test is no longer active. It then retrieves authoritative questions from MongoDB, including the stored correct answer and marks.

The controller constructs a lookup map from question identifier to authoritative answer and marks. For every submitted answer, it compares the submitted text with the stored answer after trimming whitespace. Correct responses increment the correct-answer count and add the corresponding marks to the score. This score is therefore calculated by the server from database-held answers rather than accepted from the client.

After deterministic evaluation, the controller sends the questions and submitted answers to `getAiAnalysis`. The resulting AI feedback is stored in the `AttemptTest` record together with answers, `correctAnswers`, score, submission status, and `endAt`. This sequence creates a clear research boundary: score generation is deterministic application logic, while feedback generation is probabilistic language generation.

IX. AZURE OPENAI FEEDBACK PIPELINE

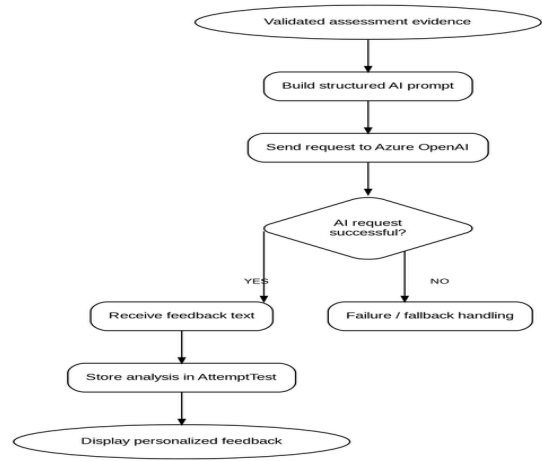


Fig. 4. Azure AI Foundry feedback workflow.

The backend configures the OpenAI SDK with an Azure AI Foundry endpoint, deployment name, API version, and API key loaded from environment variables. The `getAiAnalysis` function calls `chat.completions.create` using the configured Azure AI Foundry deployment, a system instruction, the submitted assessment evidence, temperature 0.7, and a maximum completion token limit. The system instruction defines the AI as an educational mentor and performance analyst. It receives questions containing correct answers, topics, subjects, and difficulty levels together with the student's answers. It is instructed to match responses against the corresponding questions, identify performance patterns, summarize understanding, highlight strengths, identify weak areas and recurring mistakes, recommend practical improvement steps, and end with encouragement.

The prompt also imposes important output constraints: it asks the model not to mention scores, marks, percentages, counts, or statistics, not to list individual questions, and to base feedback only on the supplied data. These constraints reduce the risk that the learner-facing feedback becomes a repetition of the numerical result.

The current implementation returns a failure string when the AI request throws an error. For a research-grade system, a structured fallback state should distinguish service failure from a valid generated report, and logs should preserve model version, prompt version, latency, and request outcome without exposing sensitive candidate data.

X. ADMINISTRATIVE WORKFLOW

The backend provides administrator operations for creating tests, adding questions, directly or from Excel files, changing test activating/deactivating tests, updating questions, retrieving conducted tests, generating reports, and downloading question-entry templates. Multer handles uploaded files and ExcelJS/XLSX support spreadsheet processing.

This administrative layer is relevant to institutional deployment because it provides a mechanism for test authors to create controlled question sets while students consume only the assigned assessment. The separation between student and administrator route groups also reduces accidental exposure of management functionality.

XI. END-TO-END WORKFLOW

- User authenticates through password/OTP or supported Google authentication.
- Backend validates the session using JWT middleware and role information.
- Authorized candidate requests available assessments.

- Candidate opens the controlled test environment and grants camera permission.

- Assessment begins after the browser enters the required test state.

- Camera recording and browser-level integrity controls operate during the attempt.

- Candidate submits answers with the attempt identifier.

- Backend validates the attempt and active test state.

- Backend retrieves authoritative question answers and marks from MongoDB.

- Server calculates correct answers and score.

- Validated questions and candidate answers are supplied to the AI analysis service.

- Azure AI Foundry generates evidence-grounded textual feedback.

- Attempt record stores answers, score, submission state, timestamps, and AI analysis.
- Frontend displays result and AI feedback through dedicated views.

IX. EMPIRICAL ASSESSMENT RESULTS

The deployed assessment was evaluated using the submitted records contained in three institutional test reports. The dataset comprises **95 students**: 44 second-year, 44 third-year, and 7 fourth-year participants. Across all records, the mean score was **65.30%**, the median was **66.67%**, and scores ranged from **0.00% to 92.86%**. The aggregate correctness rate was 66.06% (1783 correct responses from 2699 questions).

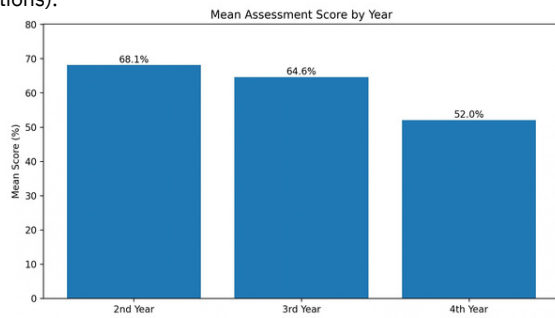


Fig. 4. Mean assessment score by academic year.

Year-wise analysis shows mean scores of 68.10% for second-year students, 64.62% for third-year students, and 52.00% for fourth-year students. These values provide descriptive evidence of performance variation across cohorts but do not by themselves establish differences in learning effectiveness.

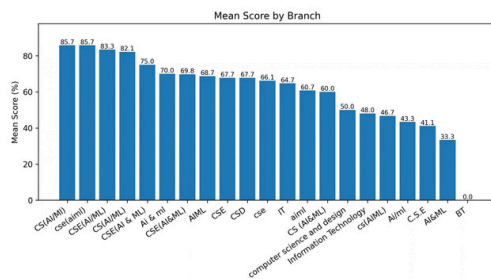


Fig. 7. Mean score by reported branch.

Score distribution

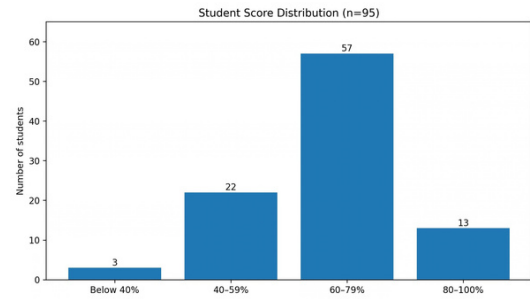


Fig. 5. Distribution of scores across all 95 submitted records.

The score distribution contained 3 students below 40%, 22 between 40% and 59%, 57 between 60% and 79%, and 13 at or above 80%. Thus, most submitted records fall in the 60–79% range, while a smaller group achieved scores of 80% or higher.

Participant-level scores

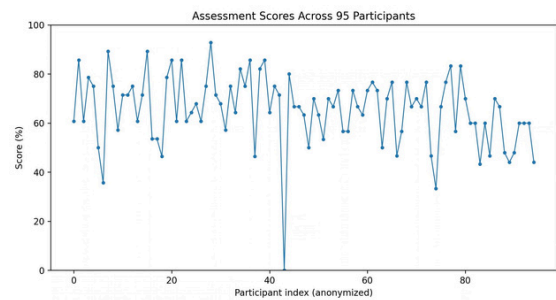


Fig. 6. Participant scores using anonymized indices.

Interpretation

The empirical records demonstrate that the implemented platform can collect completed assessments, compute server-authoritative scores, and retain structured outcomes suitable for downstream AI analysis. The results are descriptive of the submitted assessment session and should not be interpreted as a controlled experiment proving that AI feedback caused score improvement.

XII. SECURITY THREAT MODEL

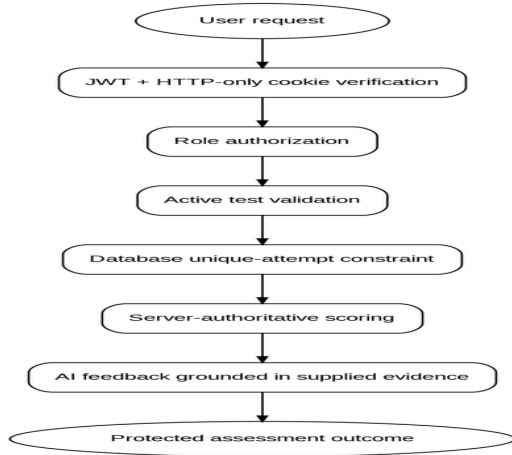


Fig. 5. Layered assessment-integrity controls.

The principal threats are grouped into five classes. T1 is unauthorized access to candidate or administrative routes. T2 is manipulation of client-side assessment state, including fabricated scores or unauthorized submission state. T3 is assessment assistance or content leakage during a remote attempt. T4 is exposure or misuse of recorded camera evidence. T5 is AI-generated feedback that contradicts or exceeds the assessment evidence.

The implementation mitigates T1 through JWT authentication and role-aware middleware, T2 through server-side score calculation and the database unique attempt constraint, and partially addresses T3 through camera recording, full-screen operation, tab-switch tracking, and interaction restrictions. T4 requires explicit retention, access-control, and deletion policies, which must be validated in the deployed infrastructure. T5 is mitigated by supplying the model with structured evidence and constraining the instruction, but must be empirically measured.

A publication should report security results as control effectiveness under a defined threat model. It should not claim that a browser application can guarantee prevention of all cheating.

XIII. PRIVACY AND ETHICAL CONSIDERATIONS

The use of camera recording changes the privacy profile of the platform. Candidates should be informed before device access is requested, the purpose of recording should be stated, and institutional policy should define who can access the recording, how long it is retained, and how it is deleted. The research dataset should avoid retaining raw video unless it is necessary for the research question and appropriate consent has been obtained.

The backend's environment-variable configuration is a positive architectural choice for server-side secrets: Azure AI Foundry credentials, database connection strings, JWT secrets, mail credentials, and Google OAuth identifiers are intended to be supplied through environment configuration rather than hard-coded values. The frontend repository previously inspected contains an AI-key placeholder; if AI requests are performed in the browser in any deployment variant, the secret must be moved to a backend-controlled service. AI feedback should be presented as assistance rather than an unquestionable judgment. If the platform is used for high-stakes recruitment or academic decisions, human review and an appeal mechanism should be provided. Fairness, accessibility, privacy, and transparency should be evaluated alongside technical performance.

XIV. RESEARCH QUESTIONS AND HYPOTHESES

RQ1: What is the reliability of the full-stack assessment workflow under normal and adverse session conditions?

RQ2: To what extent does AI feedback remain consistent with validated assessment evidence?

RQ3: Does evidence-grounded AI feedback improve subsequent technical-assessment performance compared with score-only reporting?

RQ4: How do camera-monitoring requirements affect candidate usability, privacy perception, and acceptance?

H1: Candidates receiving evidence-grounded AI feedback will show higher normalized learning gain on a comparable post-assessment than candidates receiving score-only feedback.

H2: Expert evaluators will rate evidence-grounded AI feedback higher for relevance, specificity, and actionability than generic template feedback.

H3: Greater perceived transparency about monitoring and data handling will be positively associated with acceptance of the assessment environment.

XV. EXPERIMENTAL METHODOLOGY

A comparable study should recruit participants with technical-assessment backgrounds and randomly assign them to at least a control and an experimental condition. The control group receives numerical results and conventional result information. The experimental group receives the same result plus AI-generated feedback. A stronger design adds a third condition using deterministic template feedback.

Participants should complete a pre-test, receive their assigned feedback, follow a standardized preparation interval, and complete a post-test covering comparable concepts but not identical questions. The primary dependent variable should be normalized learning gain. Secondary measures should include perceived usefulness, actionability, personalization, satisfaction, and time spent reviewing feedback.

AI feedback should be evaluated independently by subject-matter experts. Reviewers can rate factual consistency, evidence grounding, relevance, specificity, actionability, clarity, and harmful or unsupported claims. Inter-rater agreement should be reported. Statistical analysis should report sample size, descriptive statistics, effect sizes, confidence intervals, and an appropriate significance test.

Security evaluation should execute controlled test cases such as unauthorized route access, invalid/expired session tokens, duplicate attempt creation, submission after test deactivation, altered client-side score values, repeated submission, full-screen exit, tab switching, and camera-permission denial. Each test should have an expected security property and a measurable pass/fail result.

XVI. EVALUATION METRICS

XVII. DISCUSSION

The full-stack implementation provides a stronger basis for research than a frontend-only prototype because the backend establishes the authoritative assessment state. The unique database constraint, server-side answer comparison, active-test validation, and role-specific middleware demonstrate that important assessment decisions are not delegated solely to the browser.

The AI feedback design is also suitable for controlled research because it consumes explicit assessment evidence. The research can therefore measure whether generated feedback is grounded in the supplied questions and answers rather than merely asking participants whether they 'liked' the AI.

At the same time, the platform illustrates why secure assessment should be treated as a layered system. Camera recording, full-screen enforcement, tab-switch detection, and interaction restrictions each cover only part of the threat surface. Stronger institutional deployment would require server-side session controls, audit logging, rate limiting, hardened cookie settings, secure storage, retention policies, and a documented incident-response process.

The platform's AI integration should likewise be evaluated as an assistant. Its current instruction intentionally avoids displaying numerical statistics and focuses on strengths, weaknesses, recurring mistakes, and practical improvement. This makes the output more educational, but it does not by itself establish factual accuracy. Expert review and evidence-grounding measurement are required.

This manuscript's implementation-grounded but not yet an empirical results paper. The source repositories establish the software architecture and implemented mechanisms, but they do not provide controlled participant data, benchmark datasets, statistical results, or independent security certification.

The current backend AI implementation depends on an Azure AI Foundry deployment configured through environment variables. Model version, deployment configuration, prompt version, and operational cost should be recorded in the experimental setup so that results are reproducible. The AI service currently returns a failure message when the model request fails; production research should use explicit error states and a tested fallback.

The frontend test environment demonstrates camera capture with audio disabled. Therefore, the paper does not claim that the submitted implementation currently records microphone audio. If microphone support is added, the privacy model and experimental evaluation must be updated.

Browser-level controls are also limited by the security boundary of the client. Finally, no claim is made that AbhyasAI improves learning or prevents cheating until those hypotheses are tested with controlled experiments.

Future work should establish a complete empirical dataset from real or consented candidate sessions. The dataset can support longitudinal analysis of topic mastery, repeated-attempt improvement, and feedback adherence. A retrieval-augmented feedback architecture can ground recommendations in a curated technical knowledge base and institutional learning resources.

Adaptive assessment can select question difficulty according to demonstrated ability. Coding assessment can be extended with complexity analysis, test-case coverage, code-quality indicators, and debugging behavior. Interview analysis can incorporate transcripts and structured communication measures, but

multimodal inference should be introduced only after careful fairness, privacy, and accessibility validation.

Security research can extend the threat model to browser automation, multiple-device assistance, session replay, API abuse, and compromised client environments. Privacy research can investigate candidate acceptance under different disclosure and consent designs.

XX. CONCLUSION

This paper presented an implementation-grounded full-stack architecture for

AbhyasAI, combining technical assessment, secure session server-authoritative scoring, and Azure AI Foundry-based performance feedback. The frontend provides the controlled candidate experience, while the Node.js/Express backend and MongoDB database maintain the authoritative assessment state. Authentication and role-specific middleware protect student and administrator workflows. The assessment controller validates attempts, evaluates answers against database-held correct responses, calculates scores, and stores the result together with AI analysis.

The central research contribution is the separation of deterministic evidence from generative interpretation. This architecture allows the AI layer to explain and personalize validated assessment outcomes without making the language model the sole authority for scoring. It also creates a measurable research boundary for evaluating AI grounding, feedback quality, learning impact, and system reliability.

For Scopus-oriented publication, the next step is not additional feature description but rigorous experimentation. A study comparing score-only, template-based, and AI-assisted feedback; combined with controlled security testing, expert evaluation, and longitudinal learning measurement; can provide the quantitative evidence required to support the paper's research claims.

ACKNOWLEDGMENT AND AI DISCLOSURE

This manuscript was prepared with AI assistance for drafting and organization. Implementation-specific claims were derived from inspection of the submitted AbhyasAI frontend and InterviewAnalyzer backend source archives and should be verified against the deployed production configuration before submission. The final manuscript should follow the target venue's current requirements for disclosure of AI-assisted content.

REFERENCES

[1] AbhyasAI, "AI-powered exam preparation and technical interview platform," 2026. Available: <https://abhyasai.app/>

[2] S. M. et al., "Digital proctoring in higher education: A systematic literature review," International Journal of Educational Management, 2023, doi: 10.1108/IJEM-12-2022-0522.

[3] A. Marano et al., "What is the student experience of remote proctoring? A pragmatic scoping review," Higher Education Quarterly, 2024, doi: 10.1111/hequ.12506.

[4] Y. Jiao and Q. Wang, "Large language models for formative feedback in writing instruction: A systematic review of classroom interventions, feedback quality, and student outcomes," Frontiers in Education, vol. 11, 2026, doi: 10.3389/educ.2026.1834085.

[5] U. Maier, M. Seibold, and C. Klotz, "LLM-generated formative feedback in education: A qualitative systematic literature review," Computers and

Education Open, vol. 10, 2026, Art. no. 100374, doi: 10.1016/j.cao.2026.100374.

[6] J. Hattie and H. Timperley, "The power of feedback," Review of Educational Research, vol. 77, no. 1, pp. 81-112, 2007, doi: 10.3102/003465430298487.

[7] IEEE, "Artificial Intelligence (AI)-Generated Content," IEEE Author Center, accessed 2026.

IMPLEMENTATION AND VALIDATION STATEMENT

AbhyasAI was developed by the first author. The production frontend is deployed on Vercel and the backend is deployed on Render, with the application exposed through the AbhyasAI domain. Azure AI Foundry is used to provide the AI API integration for post-test result analysis and report generation. The generated report is intended to synthesize the candidate's test result into an overall assessment, strengths, weaknesses, areas for improvement, and related recommendations. The developer reports functional security validation across the implemented assessment controls, including unauthorized access, authentication manipulation, cross-user result access, duplicate submission, direct API submission, tab switching, fullscreen exit, camera denial, and refresh-related assessment behavior. These findings demonstrate functional robustness of the implemented controls; they should not be interpreted as proof of formal security certification or resistance to all possible attacks. A future controlled study should quantify false positive/negative rates, attack-detection coverage, latency, AI-feedback quality, and user experience.

REFERENCES

[1] AbhyasAI, "Technical assessment and AI-assisted feedback platform," 2026. Available: <https://abhyasai.app/>

[2] J. Hattie and H. Timperley, "The power of feedback," Review of Educational Research, vol. 77, no. 1, pp. 81-112, 2007, doi: 10.3102/003465430298487.

[3] E. Marano, P. M. Newton, Z. Birch, M. Croombs, C. Gilbert, and M. J. Draper, "What is the student experience of remote proctoring? A pragmatic scoping review," Higher Education Quarterly, vol. 78, no. 3, pp. 1031-1047, 2024, doi: 10.1111/hequ.12506.

[4] S. Han, S. Nikou, and W. Yilma Ayele, "Digital proctoring in higher education: A systematic literature review," International Journal of Educational Management, vol. 38, no. 1, pp. 265-285, 2024, doi: 10.1108/IJEM-12-2022-0522.

[5] Microsoft, "Azure AI Foundry documentation," Microsoft Learn, accessed 2026.

[6] IEEE, "IEEE Article Templates," IEEE Author Center, accessed 2026.

[7] Elsevier, "Scopus content policy and selection," Scopus, accessed 2026.

AUTHOR CONTRIBUTIONS

Anshika Singh: system conception, software design and implementation, frontend and backend development, AI integration, deployment, security validation, and manuscript preparation. Neha Singh: academic supervision, research guidance, review of methodology, and manuscript supervision.

ACKNOWLEDGMENT

The authors acknowledge R.R. Institute Of Moderna Technology, Lucknow, for the academic environment and supervision supporting the development of this work.