International Journal of Engineering Trends and Applications (IJETA) – Volume 13 Issue 3, May-Jun 2026

RESEARCH ARTICLE OPEN ACCESS

INVIGIL: A LAN-Based Offline Examination System

Akanksha Tiwari[1],Ojas Joshi[2], Rishabh Jain[3], Sachin Dixit[4], Prateek Jangra[5]
[12345]Department of Computer Science and Engineering, Global Institute of Technology, Jaipur, Rajasthan, India

ABSTRACT: The widespread adoption of digital assessments has introduced a persistent challenge: maintain- ing the integrity of examinations conducted over the internet, where candidates have unrestricted access to on- line resources. This paper presents the design and development of a LAN-Based Offline Examination System, a desktop application that enables institutions to conduct fully digital, multiple-choice examinations across multi- ple devices on a local area network, without requiring any internet connectivity. The system eliminates the pri- mary vector for online cheating by operating in a completely offline, intranet-only environment. Built using Node.js, React.js, and Express.js, and packaged into a standalone desktop executable via PKG, the application follows a client-server model with a password-protected administrator interface and a browser-based student in- terface. Testing confirms reliable multi-device operation, sub-200ms API response times, and peak memory con- sumption below 180 MB over a 45-minute exam session.

Keywords — offline examination, LAN-based testing, desk- top application, Node.js, React.js, Express.js, PKG, secure assessment, local area network, examination integrity.

1. Introduction

The transition from paper-based to digital examinations has been one of the more visible shifts in educational administration over the past decade. Online examination platforms have made it possible to conduct assessments at scale, but they carry a significant and often underestimated vulnerability: the same internet connection that delivers the exam to the candidate also provides unrestricted access to search engines, answer databases, and peer-communication tools. Monitoring software and browser-lock solutions offer partial mitigations, but technically adept students can and do find workarounds.

A more reliable approach is to remove internet access at the infrastructure level altogether. If the examination software operates exclusively on a local area network, no browser-based restriction policy is necessary there is simply no internet to exploit. This insight forms the core motivation for the system described in this paper.

We present a LAN-Based Offline Examination System developed as a final year undergraduate capstone project in Computer Science and Engineering. The application allows an institution to set up a temporary exam network using any existing Wi-Fi router or wired switch no public internet connection is required or used. One machine runs the administrator server while remaining machines connect as student clients through a web browser pointed at the server's local IP address. The administrator side is protected by password authentication to prevent unauthorized interference.

The system is built using Node.js, React.js, and Express.js, and is compiled into a portable desktop executable using PKG. This packaging approach allows the application to be launched on any compatible machine without installing Node.js or any dependencies separately, which is a signifi-cant practical advantage for deployment in institutional set- tings where system administration access may be limited.

2. Related Works

A. *Digital Examination Platforms and Their Limitations*

Internet-based examination platforms such as Google Forms, Microsoft Forms, and proprietary LMS quiz modules are widely deployed in academic institutions. While

these plat- forms provide convenience, they fundamentally depend on an active internet connection. Research documents that internet-dependent digital exams create significant academic integrity challenges, with candidates able to access external material unless strict proctoring is employed [1]. Remote proctoring solutions introduce privacy concerns and require additional hardware, increasing cost and complexity.

B. *Browser Lockdown and Proctoring Approaches*

Safe Exam Browser (SEB) and similar lockdown browser tools attempt to restrict candidate behaviour by disabling navigation, copy-paste, and screen capture functions during an online exam session [2]. However, these solutions still re- quire internet connectivity to serve the examination and re-main vulnerable to second-device attacks. Furthermore, deployment of lockdown browsers across institutional lab ma- chines requires administrative privileges and regular maintenance. Our approach sidesteps these issues by making inter- net access structurally unavailable during the exam.

C. *LAN and Intranet-Based Examination Systems*

A number of research prototypes have explored intranet- based examination. Patil and Bhagat [3] proposed a LAN- based MCQ system using PHP and MySQL, demonstrating that client-server architecture on a local network is a viable model for institutional assessments. Our work advances this direction by using a modern JavaScript stack and, critically, packaging the application as a self-contained desktop binary that requires no separate web server installation, addressing a main deployment barrier identified in prior work.

D. *Desktop Application Packaging with Node.js*

The PKG tool enables Node.js applications to be compiled into a single executable binary that bundles the Node.js run- time, all dependencies, and application code [5]. Electron.js is an alternative but adds significant application weight due to its bundled Chromium instance [6]. For a server-side application such as ours, PKG provides a leaner solution, producing an executable that launches the Express.js backend without requiring any prior Node.js installation on the host machine.

3. System Overview

The LAN-Based Offline Examination System is a client- server desktop application designed to operate entirely within a local area network. The administrator node runs the pack- aged desktop executable, which starts an Express.js server on the local machine. Student machines connect to this server by entering the administrator machine's local IP address in any standard web browser, accessing the React.js-based ex- amination interface served directly from the backend.

At no point during the operation of the system is an external internet connection required or used. The architecture ensures that all data questions, submissions, and results resides within the local network boundary

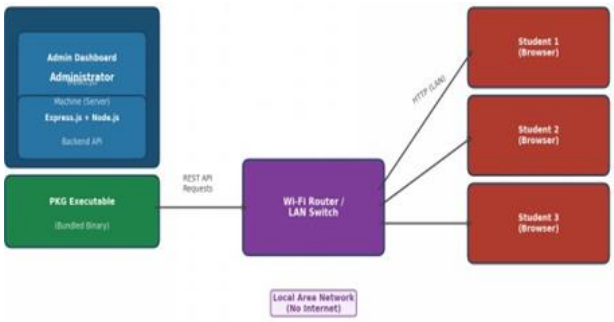

Figure 1. LAN-Based Offline Exam System Architecture Diagram

A. *Architecture and Components*

1. **Administrator Backend (Express.js + Node.js):** The core server process handles authentication, question storage, exam session management, and result aggregation, exposing a REST API consumed by both the admin dashboard and the student exam interface.
2. **Admin Dashboard (React.js):** A password-protected frontend panel

through which the administrator adds questions, starts and stops exam sessions, monitors connected students, and views results.

3. **Student Exam Interface (React.js):** A browser- accessible interface served from the local server. Con- nected students view questions, select answers, and submit responses. A countdown timer triggers automatic submission at time expiry.
4. **Desktop Executable (PKG):** The entire backend application, including the Node.js runtime and all npm dependencies, is compiled into a single executable binary distributed to the invigilator's machine.

B. Data Flow and Ingestion Modes

1. **Setup Phase**—The administrator launches the executable, logs in, and configures the examination with questions, answer options, correct answers, and time limits. All data is stored in memory during the session no external database server is required

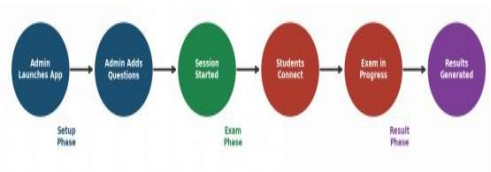

Figure 2. Examination Session Data Flow

2. **Examination Phase**—The administrator starts the session. Students connect via browser, enter their registration details, and begin the exam. Responses are submitted to the Express.js server in real time, and the administrator monitors submission status from the dashboard.

4. Implementation Methodology

A. Technology Stack Selection

The choice of technology stack was driven by three requirements: buildable by a small undergraduate team, packaging cleanly into a desktop executable, and running acceptably on typical institutional hardware. The JavaScript ecosystem satisfied all three criteria.

Node.js provides a performant, non-blocking I/O runtime well suited to handling concurrent student connections. Ex- press.js was selected as the backend framework due to its minimal abstraction overhead and large middleware ecosystem. React.js was used for both the admin dashboard and the student interface. PKG was selected over Electron.js because the application does not require a bundled browser window the student and administrator interfaces both run in the local browser, ma king Chromium overhead unnecessary.

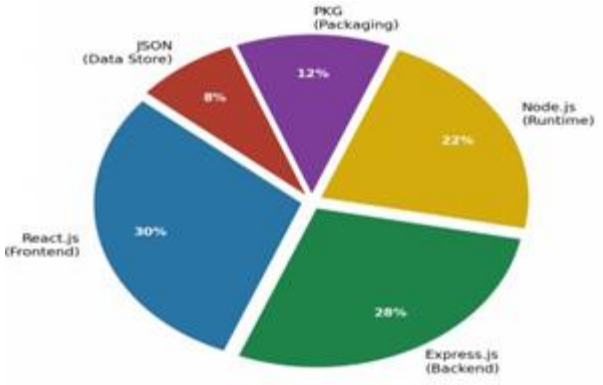

Figure 3. Technology Stack Component Distribution

B. Backend Design and API Structure

The Express.js backend implements a RESTful API organized around four resource domains: authentication, exam configuration, session management, and results. Authentication is handled through server-side session management with HTTP-only cookies. Student-facing routes reject requests outside an active exam session, preventing pre-exam ques- tion scouting.

Exam configuration endpoints accept question data as JSON objects containing the question text, four answer op- tions, and the correct answer index. Questions are held in a runtime data structure for the duration of the session and served to students only after the session is explicitly started by the administrator.

C. Frontend Implementation

The React.js frontend is structured into two

logical applications distinguished by route: the administrator dashboard at a protected route and the student exam interface at the root route. The student interface presents one question at a time with four radio-button answer options. A countdown timer component tracks remaining time and triggers automatic sub- mission at expiry. State management uses React's built-in use State and use Effect hooks without additional libraries.

D. PKG Packaging and Distribution

PKG compiles the Node.js application by traversing the de- pendency tree from the entry point and bundling all required modules with the Node.js binary into a single executable. The React frontend is built separately using the standard Cre- ate React App process, producing a static directory served by Express.js static file middleware. Both the static build directory and all Express.js dependencies are included in the Express.js server processed all simultaneous submission requests without error or data loss, with all API response times remaining below 200 milliseconds.

D. Memory Usage During Exam Session

Memory usage of the desktop executable was monitored throughout a simulated 45-minute exam session. Peak memory consumption remained below 180 MB, well within the capabilities of typical institutional hardware. A brief spike is observed at the point of mass simultaneous submission, which settles immediately after all submissions are processed.

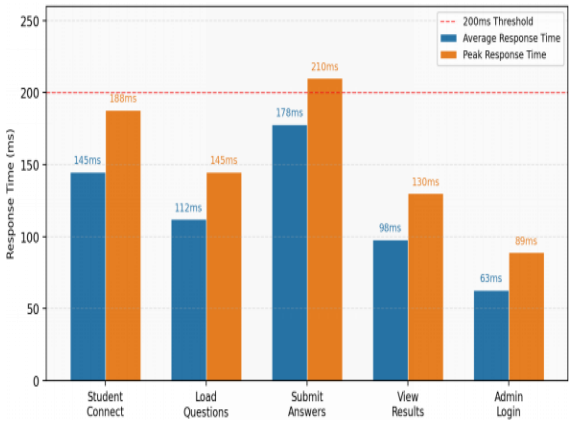

Figure 4. API Response Time Performance (4 Concurrent Clients)

5. Testing And Results

A. Test Environment

The system was tested in a controlled local area network environment comprising one administrator machine and four student machines connected to a common Wi-Fi router with no internet uplink. All machines ran Windows 10. The ex- amination used for testing consisted of 30 multiple-choice questions with a 45-minute time limit.

B. Functional Testing

Table 1. Functional Test Case Results

Test Case	Expected Behavior	Observed Result	Status
Admin login (correct password)	Dashboard accessible	Loaded successfully	Pass
Admin login (wrong password)	Access denied	Error message shown	Pass
Student connection	Interface loads	Loaded in < 2 sec	Pass
Question display	One at a time	Correct display	Pass
Timer countdown	Auto-submit at zero	Triggered correctly	Pass
Manual submission	Score computed	Score accurate	Pass
Admin views results	All submissions listed	4 results visible	Pass
Offline operation	Full functionality	Confirmed offline	Pass

C. API Performance Testing

Concurrent access performance was assessed by simulating all four student machines

submitting answers simultaneously.

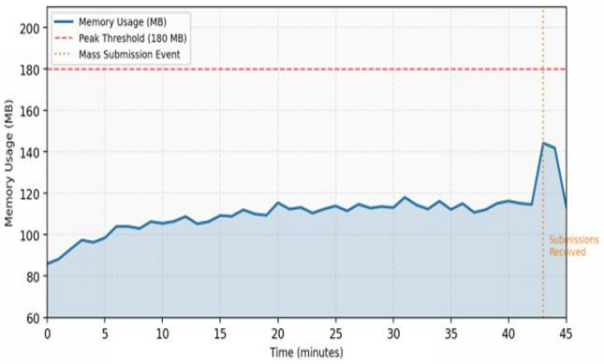

Figure 5. Server Memory Usage During 45-Minute Exam Session

D. Comparative Evaluation

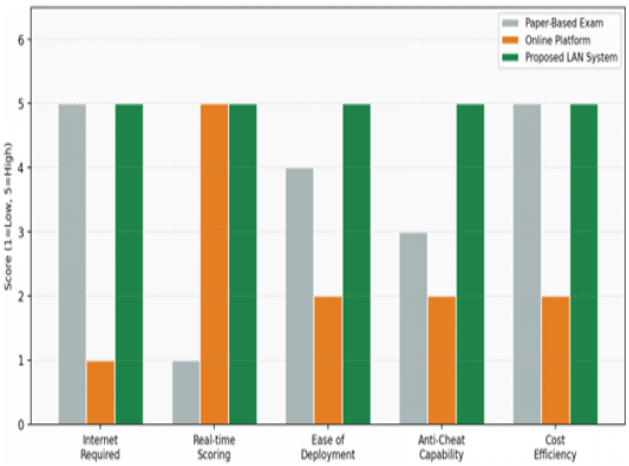

Figure 6. Comparative Evaluation of Examination Approaches (Score: 1=Low, 5=High)

Table 2. Comparison with Existing Examination Approaches

Parameter	Paper-Based	Online Platform	Proposed System
Internet Required	No	Yes	No
Real-time Scoring	No	Yes	Yes
Deployment Complexity	Low	High	Low
Cheating Risk (Online)	None	High	None
Result Access	Manual	Immediate	Immediate
Setup Cost	Low	Medium–High	Low

6. Conclusion And Future Scope

A. Conclusion

This paper presented a LAN-Based Offline Examination Sys- tem that addresses the core limitation of internet-dependent digital examination platforms. By operating entirely within a local area network with no internet connectivity, the sys- tem structurally eliminates the most common vector for technology-enabled cheating in digital assessments.

Implemented using Node.js, React.js, and Express.js and packaged as a portable desktop executable via PKG, the sys- tem substantially reduces deployment friction. Functional and performance testing confirmed correct operation across a multi-device LAN environment, reliable concurrent sub- mission handling, sub-200ms response times, and memory consumption well within institutional hardware limits.

B. Future Scope

1. **Scalability**—Integration of a lightweight embedded database such as SQLite to support larger candidate pools and persist data across longer sessions.
2. **Question Bank Management:**—Structured question banks with category tagging, difficulty levels, and randomized question selection per student to enhance suit- ability for competitive examinations.
3. **Multimedia Questions:**—Support for image-bearing questions to serve science and engineering disciplines where diagram-based questions are common.
4. **Automated Report Generation:**—PDF result reports with statistical summaries including class average, score distribution, and per-question difficulty analysis.
5. **Cross-Platform Installer:** Platform-native installers with graphical setup wizards to further reduce deploy- ment effort for non-technical invigilators.

References

[1] I. Hussain, M. S. Cakula, and M. Iqbal, "Online Exam- ination: A Review of Security Challenges and Counter-measures," International Journal of Advanced Computer Science and Applications, vol. 13, no. 4, 2022.

[2] T. Hockly, "Safe Exam Browser: A Solution to On- line Exam Integrity," eLearning Papers, no. 28, pp. 1–6, 2012.

[3] S. Patil and R. Bhagat, "Design and Implementation of LAN-Based Online Examination System," International Journal of Engineering and Computer Science, vol. 5, no. 3, pp. 15901–15904, 2016.

[4] C. Okonkwo and A. Ade-Ibijola, "Chatbots Applications in Education: A Systematic Review," Computers and Ed-ucation: Artificial Intelligence, vol. 2, 2021.

[5] Vercel, "PKG — Package Your Node.js Project Into an Executable," GitHub Repository, 2023. [Online]. Avail- able: https://github.com/vercel/pkg

[6] GitHub Inc., "Electron: Build Cross-Platform Desktop Apps with JavaScript, HTML, and CSS," 2023. [Online].

Available: https://electronjs.org

[7] OpenJS Foundation, "Node.js Documentation," 2023. [Online]. Available: https://nodejs.org/en/docs

[8] Meta Open Source, "React – A JavaScript Library for Building User Interfaces," 2023. [Online]. Available: https://reactjs.org

[9] A. Johari, S. Gupta, R. Umrainiya, "Enhancing Bank Management Systems through Salesforce Technology", International Journal of Engineering Trends and Applications (IJETA), Vol. 11, Issue. 3, pp. 97-104, 2024.

[10] R. Ajmera and N. Saxena, "Face detection in digital images using color spaces and edge detection techniques," Int. J. of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 6, pp. 718–725, Jun. 2013.

[11] R. Ajmera and D. Dharamdasani, "Comparative study of existing food delivery app," Global Research Journal, pp. 454–463, 2022.

[12] A. Kalwar and R. Ajmera, "ARQI: Model for developing web application," Int. J. on Technical and Physical Problems of Engineering (IJTPE), vol. 13, no. 47, pp. 7–13, Jun. 2021.

[13] A. Johari, R. Sharma, A. Meena, V. Tiwari, "Advancements in Pre-Trained Language Models & Their Impact on Various NLP Tasks", International Journal of Engineering Trends and Applications (IJETA), Vol. 11, Issue. 3, pp. 201-209, 2024.

[14] A. Kalwar, R. Ajmera, and C. S. Lamba, "An empirical study in small firms for web application development and proposed new parameters," Int. J. of Innovative Technology and Exploring Engineering, vol. 8, no. 4, Feb. 2019.

[15] R. Ajmera, A. Kalwar, and C. S. Lamba, "A modern study on progressions & issues of web applications development in small firms," Int. J. of Scientific Research in Science and Technology, vol. 3, no. 8, Nov.–Dec. 2017.

[16] P. Upadhyay, K. K. Sharma, R. Dwivedi and P. Jha, "A Statistical Machine Learning Approach to Optimize Workload in Cloud Data Centre," 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), pp. 276-280, 2023.

[17] P. Jha, K. K. Sharma, B. Jain, V. Sharma, "Digital Image Encryption Using AES Algorithm", EIJO Journal of Engineering, Technology And Innovative Research (EIJO–JETIR), Vol. 4, Issue. 2, 2019.