International Journal of Science, Strategic Management and Technology
Volume **02** Issue **04** April-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

A Smart Exam Management System for Fair Seating, Live Invigilation, and Student Seat Discovery

Author Details:

Sarang S S, Prakash Vaishnaw, Sundaram Pandey
MCA, Ajeenkya Dy Patil University, Pune ,India MCA, Ajeenkya Dy Patil University, Pune ,India MCA, Ajeenkya Dy Patil University, Pune ,India
Corresponding Author Email: sarangss2255@gmail.com,pkvaishnaw1617@gmail.com,sundramp92@gmail.com

https://doi.org/10.55041/ijsmt.v2i4.232

Cite this Article: S, S. S., Vaishnaw, P. & Pandey, S. (2026). A Smart Exam Management System for Fair Seating, Live Invigilation, and Student Seat Discovery. International Journal of Science, Strategic Management and Technology, *02*(04). https://doi.org/10.55041/ijsmt.v2i4.232

Abstract—

- Handling a physical examination is not only an academic responsibility but also an operational challenge. In many colleges, seating plans are still prepared manually, attendance is recorded on paper, and malpractice reports are documented only after the examination concludes. These practices are familiar, yet they are slow, error-prone, and difficult to audit. This paper presents an Exam Management System (EMS) developed to make examination administration more structured, reliable, and transparent. The project uses a Flutter frontend for the user interface and a Node.js, Express, and MongoDB backend for data processing and storage. The application supports three user roles: administrator, teacher, and student.

Administrators manage classrooms, upload institutional data, assign teachers, generate seating plans, and export reports. Teachers monitor their halls, mark attendance, report incidents, and relocate students when required. Students log in to check their allotted classroom and seat.

The central contribution of the system is a risk-aware seating engine. Instead of placing studentsin a simple serial order, the algorithm estimates local placement risk using class similarity, roll-number closeness, past absences, incident history, and class density. It then selects the lowest-risk candidate for each seat while strongly discouraging same-class pairing on the same bench. The resulting design is practical for day-to-day use and sufficiently structured to improve fairness and traceability. The project demonstrates that examination management can be strengthened significantly through a focused, interpretable, full-stack system.

Index Terms

Exam management, seating optimization, academic integrity, anti-cheating, Flutter, Node.js, MongoDB, role-based access control.should be between 150 and 250 words and must reflect the core contribution of the paper. Please avoid citations in the abstract.

Keywords— List 4–6 relevant keywords separated by semicolons.

I. INTRODUCTION

Most discussions about examinations focus on question papers, evaluation, or results. The work that happens before the examination begins is often treated as routine administration, even though it directly affects discipline, fairness, and time management. Preparing a hall, assigning seats, allocating invigilators, and responding to unexpected behavior inside the room are all operational tasks that influence the quality of the examination itself.

In many institutions, these activities are still handled through a combination of spreadsheets, printed lists, and manual adjustments. That approach can work for small groups, but it becomes difficult to manage as the number of students, classrooms, and exam sessions grows. Staff members may spend more time correcting hall plans than supervising the actual examination. Small mistakes, such as placing close classmates together or delaying a seat change after a misconduct report, can create larger problems during the session.
This project was developed with that practical context in mind. The Exam Management System brings together classroom setup, user management, seating generation, invigilation support, and student seat discovery in one application. The design is intentionally role-based. Administrators control setup and data preparation. Teachers focus on classroom supervision. Students access only the information relevant to them. This separation keeps the system simpler to use and easier to manage.

Another key motivation behind the project is the need for more deliberate seating decisions.Traditional arrangements often follow alphabetical order, roll number order, or whatever list is available at the moment. While easy to produce, those methods do not actively reduce the possibility of copying. The seating mechanism in this project was therefore designed as a preventive control, not merely as a visual arrangement.

The main objectives of the work are listed below:

1. To build a role-based software platform for in-person examination management.
2. To automate seating generation using a risk-aware placement strategy.
3. To support bulk onboarding of students, teachers, and classrooms from spreadsheet files.
4. To give teachers practical hall-level tools such as attendance marking, incident reporting, and student relocation..

II. LITERATURE REVIEW

Traditional examination management in many institutions has relied on manual registers, spreadsheets, printed seating charts, and post-exam reporting. Prior work in academic administration has consistently shown that such approaches create duplication, delay, and avoidable human error, especially when student volume, room allocation, and invigilator coordination increase. Research on campus digitization generally argues that administrative systems become more reliable when operational records are centralized, role-restricted, and easy to retrieve in real time. This is directly relevant to your project because the system combines authentication, classroom management, student records, teacher workflows, and seat discovery into one coordinated platform rather than treating each task as a separate manual activity.

III. PROBLEM STATEMENT

The main problem addressed by this project is the lack of a unified and intelligent system for managing classroom examinations. In a manual setup, information is scattered across files and people. Student lists may exist in one spreadsheet, classroom details in another, attendance in paper registers, and misconduct notes in separate written records. Because of this fragmentation, administrators and teachers often repeat the same work in different forms.
The seating process creates an additional difficulty. Manual seating is usually optimized for speed, not for examination integrity. When students from the same class or with adjacent roll numbers are seated close together, the layout may unintentionally support communication or copying. A better system should therefore do more than digitize seatcharts. It should also improve the logic used to prepare them.

IV. SYSTEM ARCHITECTURE

The project follows a client-server architecture with clear separation between interface, application logic, and data storage.

A. Frontend Layer

The user interface is built with Flutter. This allows the project to maintain a single codebase while still supporting responsive views for multiple screens and device sizes. Separate dashboards are provided for the admin, teacher, and student roles. The frontend handles navigation, token-based session persistence, file selection for uploads, and visual presentation of hall layouts and seating data.

The choice of Flutter is useful here because the same application can present complex operational screens, such as classroom management and seating grids, in a consistent way without maintaining different codebases for every platform.

B. Backend Layer

The backend is implemented using Node.js and Express. It exposes REST endpoints for authentication, administrative functions, teacher operations, student seat retrieval, and basic documentation. More importantly, the backend contains the working logic of the system. It validates classroom data, parses spreadsheet uploads, generates seating plans, records attendance, and stores incident reports.

This structure keeps decision-making on the server side, which is appropriate because seating generation and classroom state changes should remain controlled and centrally stored.

C. Database Layer

MongoDB is used as the persistence layer through Mongoose schemas. The primary collections represent users, classrooms, attendance records, and cheating incidents. The classroom schema stores the seating plan as a nested row-bench-seat structure, which fits naturally with a document model. It also keeps metadata such as seating version, generation timestamp, anti-cheat report, and the label of the last seating action.

V. FUNCTIONAL MODULES

A. Authentication and Role-Based Access The system authenticates users with hashed passwords and JSON Web Tokens. Each user is assigned one of three roles: `admin`, `teacher`, or `student`. Role-based access control ensures that users only perform the actions relevant to their role. For instance, administrators can create classrooms and upload student data, but students cannot access those operations. This not only improves security, but also keeps the interface uncluttered for end users.

B. Classroom Configuration

Each classroom is defined by rows, benches per row, seats per bench, and total capacity. The system also stores the list of classes allowed in a particular hall. Additional fields such as seating pattern and gap are available for configuration, although the present implementation relies more on risk-aware candidate placement than on simple preset patterns.

C. Spreadsheet Upload and Data Import

Real academic environments usually work with spreadsheets, not manual single-record entry. The project therefore includes upload utilities for students, teachers, and classrooms. Uploaded rows are parsed, mapped to the required schema, checked for duplicates, and then inserted into the database. Preview support helps the administrator inspect file content before the final upload, which reduces data-entry mistakes.

D. Teacher Hall Control

Teachers receive a hall-oriented workflow rather than a generic dashboard. Once a teacher opens the assigned classroom, the system returns the current seating plan. If one does not exist yet, it is generated automatically. The teacher can mark students present or absent, report malpractice with severity and notes, and optionally move a student to a new location. This makes the system useful during the exam, not only before it.

E. Student Seat Discovery

Students log in and view their classroom, row, bench, and seat number. The application also highlights the student's exact seat inside the broader classroom layout. This reduces confusion, especially when large numbers of students enter multiple rooms at the same time.

VI. METHODOLOGY OF THE SEATING ENGINE

The seating engine is the most important technical part of the project. Its purpose is not merely to fill empty seats, but to arrange them in a way that reduces risky adjacency.

The classroom is first represented as an empty three-dimensional matrix. Students are collected into a candidate pool and placed one by one. For every open seat, the algorithm examines the remaining students and evaluates how risky each candidate would be in relation to nearby occupants. The candidate with the lowest estimated risk is selected for that position.

The present implementation calculates risk using a compact set of interpretable features:

1. Whether two students belong to the same class.
2. How close their roll numbers are when they come from the same class.
3. Recorded absence history.
4. Recorded cheating-incident severity.
5. Density of the student's class within the selected population.

These values are normalized and combined through a fixed-weight linear expression, after which a sigmoid function converts the result into a bounded score. This approach is deliberately lightweight. It behaves like a machine-learning-inspired estimator, yet it is easier to interpret and maintain than a fully trained black-box model.

The algorithm also applies a strong penalty if two students from the same class are about to share the same bench. Nearby benches and previously filled seats in the same row contribute to the local score as well. After seating is completed, the system stores an anti-cheat report containing the number of students placed, maximum capacity, unseated count, average pair risk, and same-class bench violations..

VII. DISCUSSION

The implementation reflects a practical engineering choice: the system favors interpretability and direct operational usefulness over theoretical complexity. The seating engine does not claim to solve the complete optimization problem for all possible institutional constraints. Instead, it addresses a realistic subset of problems that frequently arise in classroom examinations and produces arrangements quickly enough for routine use.

The role-based design also improves maintainability. Administrators, teachers, and students interact with different parts

of the system, but they rely on a shared data model. This reduces duplication while preserving clarity in the user experience. The backend further supports this design by allowing seating plans to be generated on demand, which prevents workflow failures when one preparatory step is missed.

VIII. IMPLEMENTATION OBSERVATIONS

One practical strength of the project is that seating generation is reusable across user flows. If an administrator has not generated seating in advance, the teacher or student endpoint can still trigger seating generation when needed. This makes the system less fragile in real operating conditions, where ideal workflow order is not always followed.

Another strong feature is immediate relocation after incident reporting. When a teacher logs a misconduct case, the same workflow can move the student to a different position. The classroom record then receives a new seating version and an updated action label. This creates a simple but useful audit trail of what changed and why.The export of seating plans and student credentials in CSV format is also an important practical decision. Educational institutions often continue to rely on spreadsheet-compatible records even when software systems are introduced. Supporting export makes the project easier to adopt in such environments.

IX. BENEFITS OF THE PROPOSED SYSTEM

The benefits of the project appear at several levels. For administrators, the system reduces repeated clerical work and centralizes data that would otherwise remain scattered. For teachers, it replaces paper-heavy supervision with a live hall view that supports quick reporting and immediate corrective action. For students, it provides a direct and unambiguous way to identify the correct seat.

The larger benefit, however, is conceptual. The project treats seating as part of examination integrity rather than as a decorative final step. Even a small, explainable risk model can generate safer placements than alphabetical or roll-number-based seating in many practical situations. That makes the system valuable not only as management software, but also as an applied academic integrity tool.

IX. BENEFITS OF THE PROPOSED SYSTEM

The benefits of the project appear at several levels. For administrators, the system reduces repeated clerical work and centralizes data that would otherwise remain scattered. For teachers, it replaces paper-heavy supervision with a live hall view that supports quick reporting and immediate corrective action. For students, it provides a direct and unambiguous way to identify the correct seat.

The larger benefit, however, is conceptual. The project treats seating as part of examination integrity rather than as a decorative final step. Even a small, explainable risk model can generate safer placements than alphabetical or roll-number-based seating in many practical situations. That makes the system valuable not only as management software, but also as an applied academic integrity tool.

X. LIMITATIONS AND FUTURE WORK

The project still has limitations. The current risk model uses fixed design weights instead of values learned from institution-specific historical data. A future version could tune these parameters using past attendance and malpractice records. The present implementation also focuses on classroom-level seating rather than full campus-wide hall scheduling. Extending the system to allocate students across multiple halls would make it useful for larger examination scenarios.

Other possible improvements include QR-based student verification, better session-level attendance modeling, printable invigilator sheets, broader analytics, and timetable integration. A formal comparative study between manual seating and the current risk-aware arrangement would also strengthen the academic contribution of the project.

XI. CONCLUSION

This paper presented a full-stack Exam Management System designed to make in-person examinations easier to organize and harder to misuse. The project combines a Flutter interface with a Node.js, Express, and MongoDB backend to support administrator, teacher, and student workflows within one connected platform. Itsmost significant contribution is a risk-aware seating engine that attempts to reduce cheating opportunities by considering both academic and behavioral signals during seat placement.

The project shows that meaningful improvement in examination management does not necessarily require a heavy or overly complex system. A focused design with bulk data import, live hall supervision, student seat discovery, structured incident recording, and interpretable seating logic can make examination operations more reliable, more transparent, and more fair.

XII. References

[1] Flutter, "Flutter documentation," [Online]. Available: https://docs.flutter.dev/

[2] Express.js, "Express - Node.js web application framework," [Online]. Available: https://expressjs.com/

[3] MongoDB, "MongoDB documentation," [Online].Available: https://www.mongodb.com/docs/

[4] Mongoose, "Mongoose ODM documentation," [Online].Available: https://mongoosejs.com/docs/

[5] IETF, "JSON Web Token (JWT)," RFC 7519, May 2015.