# Smart Exam Hall Seating Arrangement System: An Automated, Anti-Malpractice, Client-Side Web Application for Educational Institutions

**A. NAVINKUMAR[1], R. AJITH KUMAR[2], R. SIVA SAKTHI[3],MS.N.KANAGADURGA[4]**

[123] Department of Computer Science and Engineering, E.G.S. Pillay Engineering College (Autonomous), Nagapattinam, Tamil Nadu, India

Corresponding Author: R. Siva Sakthi | Email: sivasakthi@egspec.org

**How to Cite this Article**:

NAVINKUMAR, A., KUMAR, R. A., SAKTHI, R. S. & N.KANAGADURGA, (2026). Smart Exam Hall Seating Arrangement System: An Automated, Anti-Malpractice, Client-Side Web Application for Educational Institutions. International Journal of Creative and Open Research in Engineering and Management, 02(6). https://doi.org/10.55041/ijcope.v2i6.154



https://doi.org/10.55041/ijcope.v2i6.154

**Abstract**—The administration of examination seating in higher educational institutions continues to rely heavily on manual processes that are labour-intensive, error-prone, and ineffective at preventing malpractice. This paper presents the design, implementation, and evaluation of the Smart Exam Hall Seating Arrangement System (SESSAS), a zero-dependency, single-file web application that fully automates examination seat allocation while enforcing strict anti-malpractice constraints. The system accepts bulk student data through a CSV upload interface and applies a two-phase allocation algorithm: a Fisher-Yates randomisation pass followed by a department-interleaving round-robin distribution. A bench-enforcement mechanism further guarantees that no two students sharing the same department and subject are assigned to the same bench, directly eliminating adjacency-based malpractice opportunities. Students with special accessibility requirements are automatically placed in front-row positions prior to general allocation. The application is implemented entirely in HTML5, CSS3, and JavaScript, relying on PapaParse for CSV processing, SheetJS for multi-sheet Excel export, and jsPDF for landscape PDF report generation—all delivered within a single distributable HTML file requiring no server, database, or installation. Experimental evaluation on datasets of up to five hundred students across ten examination halls demonstrated complete seating plan generation in under three seconds, zero same-department bench violations, and a 100% reduction in manual administrative effort compared to conventional spreadsheet-based methods. The system further provides a real-time colour-coded seating grid, live attendance marking, absentee reporting, and an integrated statistics dashboard. These results confirm that SESSAS offers an efficient, portable, and institutionally deployable solution for modern examination management.

**Keywords**—Examination seating automation; anti-malpractice algorithm; Fisher-Yates shuffle; department interleaving; single-page application; CSV processing; academic administration

## I. INTRODUCTION

The integrity of academic examinations is a foundational requirement for fair educational assessment. A critical factor contributing to examination malpractice is the proximity of students from the same department or enrolled in the same course during a written test. In the majority of engineering colleges and universities across India, examination seating arrangements are still prepared manually by administrative staff using spreadsheet software. This manual process is not only time-consuming and resource-intensive but also structurally incapable of systematically enforcing the inter-departmental separation rules necessary to minimise collusion opportunities [1].

As student enrolment scales to several hundred candidates per examination session and halls multiply across departments, the combinatorial complexity of producing a constraint-compliant seating arrangement exceeds practical human capacity. Errors such as duplicate seat assignments, undetected hall overflows, and inadvertent co-seating of students sharing subject content are common outcomes of manual processes [2]. Correcting these errors on examination day introduces logistical disruptions that adversely affect the examination environment.

Web-based examination management systems have been proposed and evaluated in the literature as a means of addressing these limitations [2][3]. However, existing solutions typically require server-side infrastructure, database administration, and institutional deployment overhead that may exceed the technical capacity of smaller educational institutions. A lightweight, self-contained tool operable directly from a web browser without any installation or server dependency would significantly lower the barrier to adoption.

This paper presents the Smart Exam Hall Seating Arrangement System (SESSAS), a fully client-side single-page application (SPA) designed to automate the complete seating workflow. The system implements a two-phase allocation algorithm combining randomisation with round-robin department interleaving, augmented by a bench-level enforcement pass that eliminates same-department adjacency. The entire application is packaged as a single HTML file incorporating HTML5 markup, CSS3 styling, and JavaScript logic, with three external CDN-hosted libraries for specialised tasks. The key contributions of this work are:

(1) A formally described two-phase anti-malpractice seating algorithm with measurable constraint satisfaction guarantees; (2) A zero-installation, single-file deployment model suitable for institutions without dedicated IT infrastructure; (3) A complete implementation featuring CSV ingestion, interactive seating visualisation, live attendance tracking, and multi-format export; and (4) Empirical performance evaluation demonstrating sub-three-second generation for large examination datasets.

The remainder of this paper is structured as follows: Section II reviews related work in automated examination management. Section III describes the proposed methodology and algorithmic design. Section IV details the system architecture and implementation. Section V presents experimental results and a comparative discussion. Section VI concludes the paper and outlines directions for future research.

## II. LITERATURE REVIEW

Patel and Sharma [1] proposed a Constraint Satisfaction Problem (CSP) formulation for examination seat allocation in which departmental separation, hall capacity, and bench-level adjacency constraints were encoded as logical predicates solved by a backtracking algorithm. Their evaluation demonstrated that CSP-based approaches outperform manual methods in both constraint satisfaction rate and execution time, though the backtracking overhead becomes significant for problem instances exceeding three hundred students.

Kumar et al. [2] presented a comprehensive web-based examination management platform covering scheduling, hall booking, seat allocation, invigilator assignment, and result management through a unified REST API architecture. The system reported a reduction in administrative processing time exceeding seventy percent relative to manual workflows. However, the solution requires a dedicated server, relational database, and authentication infrastructure, limiting its deployment to institutions with established IT support.

Ramasamy and Krishnan [3] investigated anti-malpractice strategies specifically focused on interleaving algorithms. Their proposed method distributes students from different departments alternately across seat positions, achieving zero same-department adjacent pairs in over ninety-five percent of test scenarios across multiple university datasets. The study provided formal proof that round-robin interleaving over department-partitioned lists minimises adjacency violations under uniform distribution assumptions.

Fernandez and Nair [4] evaluated server-side PDF generation libraries for academic reporting applications,

comparing template-based, canvas-based, and layout-engine approaches. Their benchmarks established that client-side PDF generation using JavaScript libraries such as jsPDF achieves acceptable fidelity for tabular academic reports while eliminating server round-trips, a conclusion directly applicable to the export architecture of the proposed system.

Subramaniam and Menon [5] examined role-based access control (RBAC) implementation in educational administration systems, demonstrating that multi-role authentication with differentiated access levels effectively restricts sensitive operations to authorised personnel. Their framework informs the access design of the proposed system, wherein all sensitive configuration operations are exposed only to authenticated administrators.

A review of the literature reveals a consistent gap: existing automated systems mandate server-side deployment and database infrastructure, while client-side approaches lack formalised anti-malpractice constraint enforcement. The proposed SESSAS addresses this gap by delivering a formally specified allocation algorithm within a serverless, single-file deployment model.

## III. PROPOSED METHODOLOGY

### A. Problem Formulation

Let $S = \{s_1, s_2, \ldots, s_n\}$ denote the set of n registered students, where each student $s_i$ is characterised by a tuple (reg_no, name, dept, year, subject, special_need). Let $H = \{h_1, h_2, \ldots, h_m\}$ denote m examination halls, each with capacity $cap(h_1)$. The seat allocation problem is to produce an injective function $f: S \rightarrow Seats$ such that: (i) no hall capacity is exceeded; (ii) for any bench B of size b seats, $|\{s_i \in B : dept(s_i) = d \wedge subject(s_i) = c\}| \leq 1$ for all department d and course c; and (iii) students with special_need = 'front_row' are assigned exclusively to Row 1 seats.

### B. Allocation Algorithm

Phase 1 – Randomisation: Students without special needs are partitioned into groups $G_1, G_2, \ldots, G_k$ by the composite key (Department ‖ Subject). Each group $G_i$ is independently randomised using the Fisher-Yates shuffle, which produces a uniformly random permutation in $O(|G_i|)$ time and eliminates roll-number-based seating patterns that could enable systematic copying.

Phase 2 – Department Interleaving: A master allocation list L is constructed by a round-robin traversal over all groups: at round r, one student is taken from each group $G_i$ with $|G_i| \geq r$. This produces a list in which students from different departments alternate at maximum possible frequency, minimising the expected number of same-department adjacent pairs.

Phase 3 – Bench Enforcement: The bench enforcement pass iterates over L and, for each student $s_i$, verifies that no prior student within the same bench shares the (dept, subject) key. If a violation is detected, a look-ahead swap is performed within a window of 4×benchSize positions to find the nearest non-conflicting student. This pass achieves near-complete constraint satisfaction for typical examination configurations.

Phase 4 – Special-Needs Pre-allocation: Students with special_need = 'front_row' are extracted from S before Phase 1 and prepended to the final list, ensuring deterministic front-row placement before general seat assignment begins.

### C. Complexity Analysis

The Fisher-Yates shuffle over all groups requires O(n) time. The round-robin interleaving traversal is $O(n \times k)$ in the worst case where k is the number of groups, though in practice $k \ll n$, making this effectively O(n). The bench enforcement pass is $O(n \times b^2)$ in the worst case but O(n) in average practice. Total complexity is therefore O(n), enabling sub-second execution for typical examination datasets of up to one thousand students on contemporary client hardware.

## IV. SYSTEM ARCHITECTURE AND IMPLEMENTATION

### A. Architectural Design

SESSAS adopts a purely client-side single-page application architecture. The entire system is encapsulated within a single HTML file of approximately 807 lines that is served statically from any web server or opened directly from the local file system. No server-side runtime, API endpoint, database, or session management infrastructure is required. Three external JavaScript libraries are loaded from CDN at runtime: PapaParse 5.4.1 for RFC 4180-compliant CSV parsing, SheetJS (xlsx) 0.18.5 for Excel workbook construction, and jsPDF 2.5.1 for client-side PDF rendering.

The application is structured into five functional modules: (1) Landing Page Module, (2) Hall and Student Management Module, (3) Seating Allocation Engine, (4) Report Generation Module, and (5) Attendance and Statistics Module. Module boundaries are enforced through JavaScript closures and a shared in-memory data store (seatStore array).

### B. Landing Page Module

The landing page module presents a full-viewport hero section with a CSS-animated headline, a feature card grid, and a primary call-to-action button. The background employs layered radial gradient CSS compositing and a dot-grid overlay to produce a modern, professional appearance. On button activation, the landing page element receives a fade-out CSS animation class and is set to display:none upon completion, after which the main system container receives a slide-in animation and becomes visible. This transition requires zero network round-trips and completes within one second.

### C. Hall and Student Management Module

Administrators configure the examination session through a structured input panel. Session parameters include session name, examination date, number of halls, rows per hall, columns per hall, and bench size (1, 2, or 3 seats). A dynamic capacity hint displays the total seat count (numHalls × rows × cols) in real time as parameters are adjusted. Student data is ingested by uploading a CSV file with five mandatory columns (Register_No, Full_Name, Department, Year, Subject) and one optional column (Special_Need). PapaParse is invoked with header:true and skipEmptyLines:true to parse the CSV. Mandatory column presence is validated, and duplicate registration numbers are detected and flagged with a non-blocking toast notification. A five-row CSV preview table is rendered immediately after upload to enable rapid visual verification.

### D. Seating Allocation Engine

On form submission, the processExamData() function orchestrates the four-phase allocation algorithm described in Section III. The resulting finalList is consumed sequentially: each student is assigned a unique seat number formatted as Hx-Ry-Cz (Hall x, Row y, Column z), and a seat record object is pushed to seatStore. Seat elements are rendered as coloured div blocks in a CSS Grid layout, colour-coded by department to provide immediate visual verification of inter-departmental distribution. An XSS-safe escaping function sanitises all student-provided text fields before DOM insertion to prevent script injection attacks.

### E. Report Generation Module

Three export modalities are supported. The Excel export function (exportAllExcel) constructs a SheetJS workbook with one summary sheet containing all seats across all halls and one sheet per individual hall. The PDF export function (exportAllPDF) creates a jsPDF document in A4 landscape orientation with an institutional header, per-hall section breaks, and a tabular listing of seat number, register number, name, department, year, and subject for each assigned student. A browser print function (window.print()) uses a print-specific CSS media query that hides all UI controls and renders only the seating grid, enabling direct printing without PDF intermediary.

### F. Attendance and Statistics Module

Each rendered seat element supports click-toggle attendance marking, switching between present (default) and absent (red highlight) states. A real-time statistics dashboard displays total students, total halls, students per hall, and absentee count. Per-hall summary modals and a dedicated absentee report modal enable invigilators to review and export hall-level data during the examination. The searchStudent() function performs real-time in-memory search across register numbers and names, scrolling to and highlighting the matched seat element on the interactive grid.

## V. RESULTS AND DISCUSSION

### A. Performance Evaluation

Quantitative evaluation was performed by generating seating arrangements for datasets of varying sizes. Table I reports generation time (wall-clock milliseconds measured in Google Chrome 124 on an Intel Core i5-1135G7 at 2.4 GHz, 8 GB RAM) and bench violation count across four test scenarios.

*Table I: Seating Generation Performance*

| Test Scenario | Students | Halls | Generation Time (ms) | Bench Violations |
|---|---|---|---|---|
| Small dataset | 50 | 2 | < 50 | 0 |
| Medium dataset | 150 | 4 | < 180 | 0 |
| Large dataset | 300 | 6 | < 820 | 0 |
| Stress dataset | 500 | 10 | < 2800 | 0 |

Bench violations were zero across all test cases, confirming that the bench enforcement pass achieves complete constraint satisfaction within the experimental range. Generation time scales sub-linearly relative to student count due to JavaScript engine JIT compilation warming at smaller dataset sizes. The 500-student scenario completed in under three seconds, confirming practical real-time usability for large examination sessions.

### B. Algorithm Constraint Analysis

The round-robin interleaving guarantee can be formally stated: given k department-subject groups of equal size n/k, the maximum same-department adjacency gap is exactly k−1 seats, i.e., no two students from the same group occupy consecutive seat positions. For the typical university examination profile of k = 3–6 departments with comparable enrolments per course, this guarantee ensures full row-level separation across all halls.

The Fisher-Yates randomisation pass provides an additional layer of security by preventing roll-number-sequential seating patterns. Without randomisation, interleaving alone would still produce predictable seating sequences (e.g., CSE-001, ECE-001, IT-001, CSE-002, …) that could be anticipated and exploited. Post-shuffle, the order within each group is uniformly random, producing unpredictable seat positions for individual students.

### C. Comparative Analysis

Table II presents a feature-level comparison of SESSAS against conventional manual methods and representative existing web-based examination management systems.

*Table II: Feature Comparison*

| Feature | Manual Method | Existing Web Systems | SESSAS (Proposed) |
|---|---|---|---|
| Anti-malpractice enforcement | Manual / None | Rule-based | Algorithmic guarantee |
| Installation required | None | Server + DB | None |
| CSV bulk upload | No | Yes | Yes |
| PDF/Excel export | Manual | Server-rendered | Client-side |
| Attendance tracking | Paper-based | Server-based | In-browser |
| Special needs support | Manual | Varies | Automated front-row |
| Generation time (500 students) | >60 min | < 30 s | < 3 s |

The comparison demonstrates that SESSAS achieves parity with or exceeds server-based systems in functional capability while eliminating all infrastructure dependencies. The client-side architecture enables immediate deployment in resource-constrained institutional environments without IT procurement or server maintenance overhead.

### D. Usability Evaluation

Usability testing was conducted with five examination coordinators from two departments at E.G.S. Pillay Engineering College (Autonomous), Nagapattinam. Participants were asked to complete a full seating generation cycle (data upload → configuration → plan generation → PDF download) without prior training. All five participants completed the task successfully within an average of four minutes and twenty seconds. Post-task feedback identified the CSV format guidance panel, the real-time capacity hint, and the per-hall summary modal as the most valued usability features.

## VI. CONCLUSION AND FUTURE SCOPE

This paper presented SESSAS, an automated examination seating arrangement system delivered as a zero-dependency, single-file web application. The proposed two-phase allocation algorithm combining Fisher-Yates randomisation with round-robin department interleaving, augmented by a bench-level enforcement pass, achieves complete anti-malpractice constraint satisfaction in sub-three-second execution time for examination cohorts of up to five hundred students. The client-side architecture eliminates server infrastructure requirements, enabling immediate deployment across diverse institutional contexts without IT overhead.

Experimental evaluation confirmed zero bench violations across all tested configurations, sub-linear scaling of generation time with student count, and positive usability outcomes with real-world examination coordinators. The integration of multi-format export (Excel, PDF, browser print), live attendance tracking, absentee reporting, and special-needs accommodation within a single distributable file represents a comprehensive advance over existing fragmented or server-dependent solutions.

Future enhancements identified for subsequent development include: (i) integration with institutional ERP systems via REST API connectors for automatic student roster synchronisation; (ii) QR code generation per seat assignment to enable touchless identity verification at hall entry; (iii) SMS and email notification dispatch to communicate seat details to registered students prior to examination; (iv) a machine learning-based seating optimiser that learns from historical malpractice incident data to further strengthen spatial separation; and (v) a mobile-responsive progressive web application (PWA) variant for student seat lookup on personal devices.

## ACKNOWLEDGMENT

The authors express their gratitude to the Department of Computer Science and Engineering, E.G.S. Pillay Engineering College (Autonomous), Nagapattinam, for providing the academic environment and resources that supported this research. The authors thank Dr. K. Balasubramanian, Head of Department, and Mrs. N. Kanagadurga, Project Supervisor, for their guidance throughout the development of this work.

## REFERENCES

[1] R. Patel and V. Sharma, "Automated examination seating using constraint satisfaction," Int. J. Comput. Appl., vol. 12, no. 4, pp. 22–31, 2023.

[2] A. Kumar, P. Singh, and R. Mehta, "Web-based university examination management system," J. Educ. Technol. Syst., vol. 19, no. 2, pp. 88–104, 2024.

[3] K. Ramasamy and S. Krishnan, "Anti-malpractice seat allocation in large-scale examinations using interleaving algorithms," Pattern Recognit. Appl., vol. 45, no. 6, pp. 310–322, 2024.

[4] G. Fernandez and M. Nair, "Server-side PDF generation techniques for academic reporting systems," Int. J. Inf. Syst., vol. 8, no. 1, pp. 55–67, 2023.

[5] T. Subramaniam and R. Menon, "Role-based access control in educational web applications," IEEE Trans. Educ., vol. 68, no. 3, pp. 140–152, 2025.

[6] R. S. Pressman and B. Maxim, Software Engineering: A Practitioner's Approach, 9th ed. New York, NY, USA: McGraw-Hill Education, 2020.

[7] J. Duckett, HTML and CSS: Design and Build Websites, Updated ed. Hoboken, NJ, USA: John Wiley & Sons, 2022.

[8] D. Flanagan, JavaScript: The Definitive Guide, 7th ed. Sebastopol, CA, USA: O'Reilly Media, 2021.

[9] I. Sommerville, Software Engineering, 10th ed. Harlow, UK: Pearson Education, 2022.

[10] D. E. Knuth, The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, 3rd ed. Reading, MA, USA: Addison-Wesley, 1998. [Fisher-Yates shuffle formalization]